1999-09

Development and implementation of a shell element with pressure variation through the thickness and void growth and nucleation effects

McDermott, Patrick M.

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/13705

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DEVELOPMENT AND IMPLEMENTATION OF A SHELL ELEMENT WITH PRESSURE VARIATION THROUGH THE THICKNESS AND VOID GROWTH AND NUCLEATION EFFECTS

by
Patrick M. McDermott

September 1999

Thesis Advisor: Young W. Kwon

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 1999	**3. REPORT TYPE AND DATES COVERED** Mechanical Engineer's Thesis

4. TITLE AND SUBTITLE
DEVELOPMENT AND IMPLEMENTATION OF A SHELL ELEMENT WITH PRESSURE VARIATION THROUGH THE THICKNESS AND VOID GROWTH AND NUCLEATION EFFECTS

5. FUNDING NUMBERS

6. AUTHOR(S)
McDermott, Patrick M.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

A shell formulation was developed from a three-dimensional solid. The shell element has four corner nodes at which there are three displacements and three rotations as nodal degrees of freedom, and includes both transverse shear and transverse normal deformations. The element utilizes reduced integration along the in-plane axes and full integration along the transverse axis. The formulation incorporates the Gurson constitutive model for void growth and plastic deformation. An algorithm for stable solutions of the nonlinear constitutive equations is also developed. Hourglass mode control is provided by adding a small fraction of internal force determined through full integration along the in-plane axes and reduced integration along the transverse axis. Implementation into a research finite element program is discussed. Numerical examples are provided to verify the accuracy of the new element and to show the importance of the transverse normal stress, void effects on plastic strain, and the necessity of applying a drilling moment.

14. SUBJECT TERMS
Finite Element, Shell Formulation, Void Model, Elasto-plastic, High Order Element

15. NUMBER OF PAGES
124

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFI- CATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

DEVELOPMENT AND IMPLEMENTATION OF A SHELL ELEMENT WITH PRESSURE VARIATION THROUGH THE THICKNESS AND VOID GROWTH AND NUCLEATION EFFECTS

Patrick M. McDermott
Lieutenant, United States Navy
B.S.E.E., Memphis State University, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MECHANICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
September 1999

Author: [redacted]
Patrick M. McDermott

Approved by: [redacted]
Young W. Kwon, Thesis Advisor

[redacted]
M. D. Kelleher, Acting Chair
Department of Mechanical Engineering

ABSTRACT

A shell formulation was developed from a three-dimensional solid. The shell element has four corner nodes at which there are three displacements and three rotations as nodal degrees of freedom, and includes both transverse shear and transverse normal deformations. The element utilizes reduced integration along the in-plane axes and full integration along the transverse axis. The formulation incorporates the Gurson constitutive model for void growth and plastic deformation. An algorithm for stable solutions of the nonlinear constitutive equations is also developed. Hourglass mode control is provided by adding a small fraction of internal force determined through full integration along the in-plane axes and reduced integration along the transverse axis. Implementation into a research finite element program is discussed. Numerical examples are provided to verify the accuracy of the new element and to show the importance of the transverse normal stress, void effects on plastic strain, and the necessity of applying a drilling moment

TABLE OF CONTENTS

I. INTRODUCTION 1

II. FINITE ELEMENT FORMULATION 5

A. GEOMETRY 5
B. DISPLACEMENT 7
C. COORDINATE TRANSFORMATION 8
D. STRAIN DISPLACEMENT RELATION 10
E. JACOBIAN MATRIX 11
F. STRESS-STRAIN RELATIONSHIP 12
G. DRILLING MOMENTS 12
H. INTERNAL FORCE, MASS AND THEIR ASSEMBLY 14
I. EXPLICIT TIME INTEGRATION 15

III. DAMAGE CONSTITUTIVE EQUATIONS 17

A. GURSON'S VOID MODEL 17
B. IMPROVING STABILITY IN THE CONSTITUTIVE EQUATIONS 20
C. STRAIN HARDENING 22

IV. HOURGLASS MODE CONTROL 25

V. IMPLEMENTATION IN RESEARCH-ORIENTED FE CODE 31

A. PREPROCESSOR 31
B. POSTPROCESSOR 31
C. FEA SHELL ELEMENT SUBROUTINE DESCRIPTION 32

VI. NUMERICAL EXAMPLES 35

A. ELASTIC PLATE 35
B. THICK CLAMPED PLATE UNDER PRESSURE LOAD IN ELASTO-PLASTIC REGION 36
C. THICK CLAMPED PLATE WITH CENTRAL POINT LOAD IN THE ELASTO-PLASTIC REGION 42
D. SIMPLY SUPPORTED PLATE WITH CENTRAL POINT LOAD IN THE ELASTO-PLASTIC REGION 46
E. ELASTIC PINCHED CYLINDER 47
F. THICK PINCHED CYLINDER IN THE ELASTO-PLASTIC REGION 48
G. ELASTIC SPHERICAL CAP WITH A CENTER HOLE 51
H. SPHERICAL CAP WITH A CENTER HOLE, IMPACT LOADING 53
I. SPHERICAL CAP WITH A CENTER HOLE, ELASTO-PLASTIC LOADING 54

VII. CONCLUSIONS 61

APPENDIX A. PREPROCESSOR IN MATLAB FOR FEA. 63

APPENDIX B. POST-PROCESSOR IN MATLAB FOR FEA. 73

APPENDIX C. FILE LISTING FOR FEA SHELL ELEMENT SUBROUTINE. 91

LIST OF REFERENCES 107

INITIAL DISTRIBUTION LIST 111

LIST OF FIGURES

Figure 1. Element Cross Section. 5

Figure 2. Displacement Vector Orientation. 7

Figure 3. Calculating a New Yield Stress. 23

Figure 4. Hourglass Modes in a Pinched Cylinder Model. 25

Figure 5. Hourglass Mode Control Integration Points. 26

Figure 6. Pinched Cylinder Model with Hourglass Mode Control. 28

Figure 7. Outline of FEA Implementation. 34

Figure 8. Nine Element Clamped Plate Mesh. 37

Figure 9. Void Effects in a Clamped Plate with Pressure Loading: Top (a) - No Void Effects, Bottom (b) - Void Effects. 40

Figure 10. Transverse Normal Stress Variation through Shell Thickness: Clamped Plate with Pressure Loading and Void Effects. 41

Figure 11. Stress Component Time History in Bottom Fiber: Clamped Plate with Concentrated Load, Void Effects, and Drilling Moment Applied. 43

Figure 12. Porosity versus Effective Plastic Strain in Bottom Fiber: Clamped Plate with Concentrated Load, Void Effects, and Drilling Moment Applied. 44

Figure 13. Mesh Structure for Pinched Cylinder. 50

Figure 14. Stress Component Time History in Bottom Fiber: Pinched Cylinder, No Void Effects or Drilling Moment. 50

Figure 15. Stress Component Time History in Bottom Fiber: Pinched Cylinder, Void Effects and Drilling Moment Applied. 51

Figure 16. Mesh Structure for Spherical Cap. 53

Figure 17. Node One Displacement Time History: Pinched Spherical Cap with Impact Loading. 54

Figure 18. Porosity versus Effective Plastic Strain in Inner Fiber: Spherical Cap with Void and Drilling Moment Effects. 56

Figure 19. Comparison of Void and Drilling Moment Effects in a) Compression (Top) and b) Tension (Bottom) for Spherical Cap. 57

Figure 20. Contact Node Mean Displacement Time History: Spherical Cap with Void and Drilling Moment Effects. 58

Figure 21. Contact Node Displacement: Spherical Cap with 5% Greater Load. 59

Figure 22. Deformed Structure at End of Analysis: Spherical Cap with 5% Greater Load. 60

Figure 23. Screen Capture of FEA Preprocessor. 89

LIST OF TABLES

Table 1. Comparison of Results for Elastic Clamped Plate. 36
Table 2. Material Properties of Clamped Plate. 36
Table 3. Void Characteristics of Clamped Plate. 37
Table 4. Summary of Results for Clamped Plate with Pressure Load. 42
Table 5. Summary of Results for Clamped Plate Subjected to Point Force. 45
Table 6. Summary of Results for a Simply Supported Plate with a Point Force. 46
Table 7. Comparison of Results for Elastic Pinched Cylinder. 47
Table 8. Summary of Results for Elasto-Plastic Pinched Cylinder. 49
Table 9. Comparison of Results for Spherical Cap with Elastic Loading, and Results from MacNeal and Harder [26]. 52
Table 10. Summary of Results for Elasto-Plastic Spherical Cap. 56

I. INTRODUCTION

Since shell structures are efficient load-carrying structural members, they have been dominant in most structural applications. However, the finite element formulation of shells poses some difficulty. As a result, extensive study has been devoted to developing better shell elements. Because of the abundance of papers on this subject, no attempt is made here to summarize all of them. Some of the related past work is given in references [1-12].

Void growth and nucleation can have a significant effect on plastic flow [13]. Since voids act as stress concentrators, the overall effect is to reduce the stress under plastic flow, and increase the plastic strain [14-16]. The model proposed by Gurson [17] appears to have been adopted as the standard for incorporating void growth and nucleation effects into a numerical solution of elasto-plastic problems. Previous work has been done on improving the efficiency and accuracy of plasticity computations, applying plasticity to plate/shell elements, and incorporating void growth and nucleation effects in solid elements [18-20]. The element presented in this paper incorporates a modified version of the algorithm for three-dimensional solids proposed by Aravas [21] into a shell element. This shell element assumes a modified plane-stress condition, and utilizes both the hydrostatic pressure and the deviatric stress. Due to the importance of the hydrostatic pressure on void growth and nucleation, this element includes the transverse normal stress.

The algorithm proposed by Aravas is not unconditionally stable when applied to this modified plane stress condition [21]. This paper also presents modifications to the original algorithm, which greatly enhance the stability of the solution, at the cost of a slight decrease in computational efficiency. This algorithm also allows for more complicated work-hardening profiles than the typical exponent law ($\sigma = K\varepsilon^n$). Full modeling of the entire stress-strain curve is accomplished by a piece-wise linear approximation. While this method is not particularly useful for analytic approaches, it appears to be helpful in strictly numerical solutions. The stress-strain relationship may be taken directly off the results of a standard tensile test.

For thick shell applications, the transverse normal stress and strain cannot be neglected. The transverse normal stress and strain affect both the hydrostatic pressure and the deviatoric stress, which in turn affect plastic deformation and void growth and nucleation. The element presented here extends the typical use of the drilling degree of freedom (DOF), as outlined in Hughes and Brezzi [22], to incorporate the effects of transverse normal strain. The drilling DOF is important in transmitting stress and strain to elements across sharp bends and curves, and is therefore already included in a variety of shell elements [2,4,6,7,10]. In addition to this traditional usage, the present element utilizes the drilling DOF to compute the transverse normal deformation. The importance of including the transverse normal deformation is outlined in Essenburg [23].

The present study formulates a shell element for transient analysis. The element can have elasto-plastic deformation with void growth and nucleation. Gurson's void model is used as a basis for the void constitutive model. The shell element includes both transverse shear deformation and the transverse normal deformation for thick shell applications. The drilling degree of freedom is used for computing the deformation through the thickness of a thick shell. An algorithm for stable solutions of the nonlinear constitutive equations is also developed.

Some example problems are presented to evaluate the formulation and to investigate the effects of the transverse normal strain for thick shells in association with elasto-plastic deformation, including void effects. The sections that follow are: the formulation of the shell element, discussion of the elasto-plastic constitutive equation including void nucleation and growth, the stable solution algorithm including spurious (hourglass) mode control, numerical examples and discussion of results, and conclusions.

II. FINITE ELEMENT FORMULATION

A. GEOMETRY

A point in a shell structure can be expressed by a vector sum of two vectors. The first vector is a position vector from the origin of the global coordinate system to a point on a reference surface of the shell element. The second vector is a position vector from this reference surface to the point under consideration. The surface that spans the center of the transverse axis is used as the reference surface in this formulation, although any surface would suffice. The first vector terminates at the reference surface directly below the point in question. The second vector is then the normal from the reference surface that intersects the desired point. Figure 1 shows this relationship. Two shape functions are used to describe

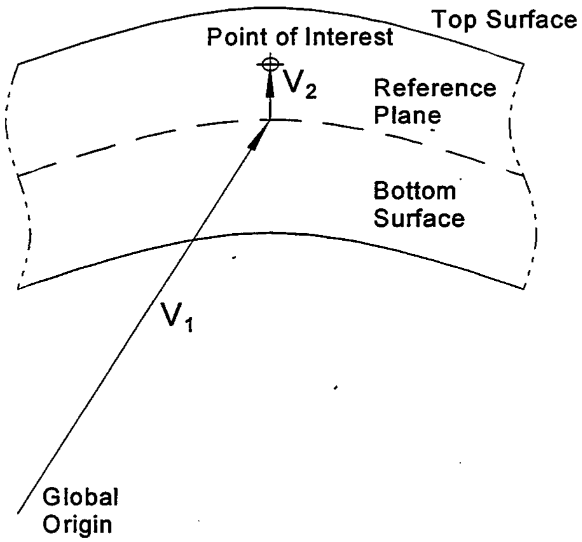


Figure 1. Element Cross Section.

a position in the element; N^k is the two dimensional shape function in the ξ-η plane, and H^k is the one dimensional shape function along the ζ axis, where (ξ,η,ζ) describes a point in the natural coordinate system. A generic point in the shell may now be described in terms of the position vectors of the nodes and the shape functions:

$$x_i(\xi,\eta,\zeta)=\sum_{k=1}^{n} N^k(\xi,\eta)\ x_i^k + \sum_{k=1}^{n} N^k(\xi,\eta)\, H^k(\zeta)\, V_{3i}^k \quad (i = 1,2,3) \tag{1}$$

where x_i^k is the position vector of node *k* in the reference surface; V_{3i}^k is the unit vector at the node *k*; and *n* is the number of nodes per element. In the present formulation, a four-node shell element is considered. The unit vector V_{3i}^k is defined as:

$$V_{3i}^k = \frac{(x_i^k)^{top} - (x_i^k)^{bottom}}{\left\|(x_i^k)^{top} - (x_i^k)^{bottom}\right\|} \tag{2}$$

where *top* and *bottom* indicate the top and bottom surfaces of the shell, and || || denotes the Euclidean norm. The one-dimensional shape function H^k is expressed as:

$$\mathrm{H}^k(\zeta) = \left[\frac{1}{4}(1+\zeta)(1-\bar{\zeta}) - \frac{1}{4}(1-\zeta)(1+\bar{\zeta})\right]\ \left\|(x_i^k)^{top} - (x_i^k)^{bottom}\right\| \tag{3}$$

in which $\bar{\zeta}$ indicates the location of the reference surface and varied from -1 to 1 ($\bar{\zeta}=0$ denotes the mid-surface). The two-dimensional shape function N^k is expressed as:

$$\begin{aligned} N^1 &= \tfrac{1}{4}(1-\xi)(1-\eta) \\ N^2 &= \tfrac{1}{4}(1+\xi)(1-\eta) \\ N^3 &= \tfrac{1}{4}(1+\xi)(1+\eta) \\ N^4 &= \tfrac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{4}$$

B. DISPLACEMENT

The displacement field in a shell can be written as:

$$u_i(\xi,\eta,\zeta) = \sum_{k=1}^{n} N^k(\xi,\eta)\, u_i^k + \sum_{k=1}^{n} N^k(\xi,\eta)\, H^k(\zeta)(-V_{2i}^k\theta_1^k + V_{1i}^k\theta_2^k + V_{3i}^k\theta_3^k) \quad (i=1,2,3) \qquad (5)$$

in which u_i is the displacement along the x_i axis, u_i^k is the nodal displacement at the node k, and unit vectors V_{1i}^k and V_{2i}^k lie along the reference surface. V_{1i}^k, V_{2i}^k and V_{3i}^k are mutually perpendicular. θ_1^k, θ_2^k and θ_3^k are rotational degrees of freedom along the unit vectors V_{1i}^k, V_{2i}^k and V_{3i}^k, respectively. The right-hand rule is assumed for the positive direction of each rotation. Figure 2 illustrates the relationship among these vectors.

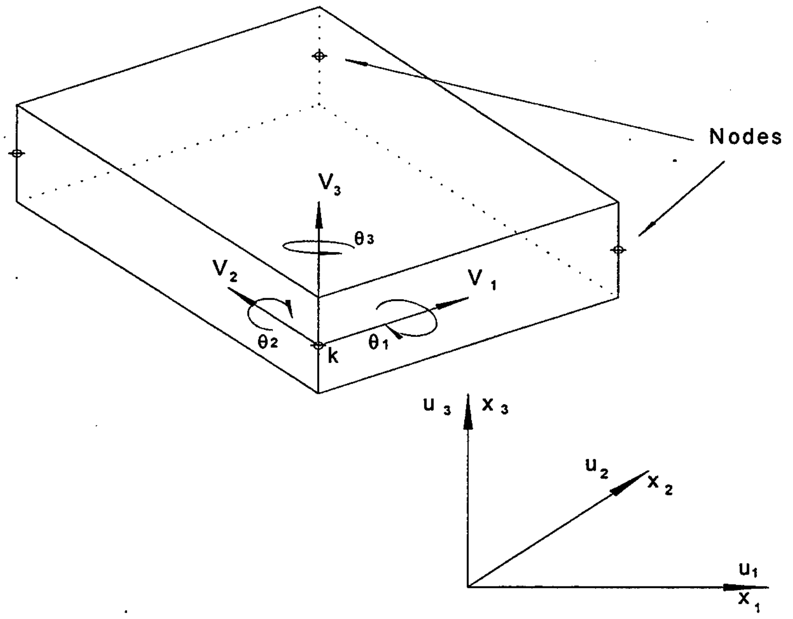


Figure 2. Displacement Vector Orientation.

θ_1^k and θ_2^k are the bending rotations, while θ_3^k is the drilling rotational degree of freedom. The role of θ_3^k can be clarified by considering a flat plate parallel to the x_1-x_2 plane. Now Eq. (5) can be rewritten for the transverse displacement as:

$$u_3 = \sum_{k=1}^{n} N^k u_3^k + \sum_{k=1}^{n} N^k H^k \theta_3^k \qquad (6)$$

Equation (6) demonstrates that the transverse deformation varies through the plate thickness (i.e. along the ζ axis) with θ_3^k. In this way, the transverse normal deformation is included in this formulation, along with the transverse shear deformations.

C. COORDINATE TRANSFORMATION

Combining the three unit direction vectors into matrix $[\mathbf{T}_p]$ provides the rotation transformation matrix, or matrix of direction cosines, as shown in Eq. (7). $[\mathbf{T}_p]^{-1}$ is used to transform the nodal degrees of freedom from the global coordinate system to local coordinates, as shown in Eq. (8), where *k* is the node number. The components of *{d}* at the four nodes of an element are shown in Eq. (9). Once the internal force vector is generated in local coordinates, $[\mathbf{T}_p]$ is used to transform the local vectors back into global coordinates. This procedure will be discussed later.

$$[\mathbf{T}_p] = \left| \vec{V}_1 \, \vec{V}_2 \, \vec{V}_3 \right|_{(3x3)} \qquad (7)$$

$$\left\{d^k\right\}_{(6x1)} = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & & & \\ 0 & 0 & 0 & & [T_p]^{-1} & \\ 0 & 0 & 0 & & & \end{vmatrix} \left\{d^k_{global}\right\}_{(6x1)} \quad (k=1,2,3,4) \qquad (8)$$

$$\left\{d\right\} = \left\{d^1\, d^2\, d^3\, d^4\right\}^T \qquad (9)$$

The strain transformation matrix, [T], is used to transform calculated strain from the global coordinate system to local coordinates. Transforming the resulting stress from local coordinates to the global coordinate system would normally require using $[\mathbf{T}]^{-1}$, but since [T] is orthogonal, $[\mathbf{T}]^{-1} = [\mathbf{T}]^{T}$, where $[\mathbf{T}]^{T}$ is the transpose of [T]. This property negates the requirement to invert a six by six matrix. For a detailed derivation of these transformations, refer to Cook [24]. The strain transformation matrix is explicitly defined in Eq. (10), where V_{ij} is the cosine of the direction vector V_i in the x_j direction.

$$[T] = \begin{bmatrix} V_{11}^2 & V_{12}^2 & V_{13}^2 & V_{11}V_{12} & V_{12}V_{13} & V_{11}V_{13} \\ V_{21}^2 & V_{22}^2 & V_{23}^2 & V_{21}V_{22} & V_{22}V_{23} & V_{21}V_{23} \\ V_{31}^2 & V_{32}^2 & V_{33}^2 & V_{31}V_{32} & V_{32}V_{33} & V_{31}V_{33} \\ 2V_{11}V_{21} & 2V_{12}V_{22} & 2V_{13}V_{23} & V_{11}V_{22}+V_{21}V_{12} & V_{12}V_{23}+V_{22}V_{13} & V_{13}V_{21}+V_{23}V_{11} \\ 2V_{21}V_{31} & 2V_{22}V_{32} & 2V_{23}V_{33} & V_{21}V_{32}+V_{31}V_{22} & V_{22}V_{33}+V_{32}V_{23} & V_{23}V_{31}+V_{33}V_{21} \\ 2V_{11}V_{31} & 2V_{12}V_{32} & 2V_{13}V_{33} & V_{11}V_{32}+V_{31}V_{12} & V_{12}V_{33}+V_{32}V_{13} & V_{13}V_{31}+V_{33}V_{11} \end{bmatrix} \qquad (10)$$

D. STRAIN DISPLACEMENT RELATION

The six components of the strain tensor are computed from Eq. (5) by taking its derivative with respect to the x_i axis. In matrix form, the result for a four-node element is:

$$\{\varepsilon\} = [\mathbf{B}]\{d\} \tag{11}$$

$$\{\varepsilon\} = \{\, \varepsilon_{11}\, \varepsilon_{22}\, \varepsilon_{33}\, \gamma_{12}\, \gamma_{23}\, \gamma_{13} \,\}^T \tag{12}$$

where

$$[\mathbf{B}] = \left[[\mathbf{B}^1][\mathbf{B}^2][\mathbf{B}^3][\mathbf{B}^4] \right] \tag{13}$$

The detailed expression for $[\mathbf{B}^k]$ is:

$$[\mathbf{B}^k] = \begin{bmatrix} \frac{\partial N^k}{\partial x_1} & 0 & 0 & -g_1^k V_{21}^k & g_1^k V_{11}^k & g_1^k V_{31}^k \\ 0 & \frac{\partial N^k}{\partial x_2} & 0 & -g_2^k V_{22}^k & g_2^k V_{12}^k & g_2^k V_{32}^k \\ 0 & 0 & \frac{\partial N^k}{\partial x_3} & -g_3^k V_{23}^k & g_3^k V_{13}^k & g_3^k V_{33}^k \\ \frac{\partial N^k}{\partial x_2} & \frac{\partial N^k}{\partial x_1} & 0 & -g_2^k V_{21}^k - g_1^k V_{22}^k & g_2^k V_{11}^k + g_1^k V_{12}^k & g_2^k V_{31}^k + g_1^k V_{32}^k \\ 0 & \frac{\partial N^k}{\partial x_3} & \frac{\partial N^k}{\partial x_2} & -g_k^k V_{22}^k - g_k^k V_{23}^k & g_k^k V_{12}^k + g_k^k V_{13}^k & g_k^k V_{32}^k + g_k^k V_{33}^k \\ \frac{\partial N^k}{\partial x_3} & 0 & \frac{\partial N^k}{\partial x_1} & -g_3^k V_{21}^k - g_1^k V_{23}^k & g_3^k V_{11}^k + g_1^k V_{13}^k & g_3^k V_{31}^k + g_1^k V_{33}^k \end{bmatrix} \tag{14}$$

in which

$$g_i^k = \frac{\partial N^k}{\partial x_i} H^k + N^k \frac{\partial H^k}{\partial x_i} \tag{15}$$

The vector $\{d^k\}$ is defined as:

$$\{d^k\} = \{\, u_1^k\, u_2^k\, u_3^k\; \Theta_1^k\, \Theta_2^k\, \Theta_3^k \,\} \tag{16}$$

where u_i is the displacement along the x_i direction at node k, and Θ_i^k is the rotational displacement about the global x_i axis at node k. The matrix $[\mathbf{B}^k]$ must be calculated for each integration point.

E. JACOBIAN MATRIX

Computing the derivatives $\frac{\partial N^k}{\partial x_i}$ and $\frac{\partial H^k}{\partial x_i}$ requires the Jacobian matrix, defined as

$$[\mathbf{J}]=\begin{bmatrix} x_{1,\xi} & x_{2,\xi} & x_{3,\xi} \\ x_{1,\eta} & x_{2,\eta} & x_{3,\eta} \\ x_{1,\zeta} & x_{2,\zeta} & x_{3,\zeta} \end{bmatrix} \tag{17}$$

where

$$x_{i,\xi}=\frac{\partial x_i}{\partial \xi}=\sum_{k=1}^{n}\frac{\partial N^k}{\partial \xi}x_i+\sum_{k=1}^{n}\frac{\partial N^k}{\partial \xi}H^k V_{3i}^k \quad (i=1,2,3) \tag{18}$$

$$x_{i,\eta}=\frac{\partial x_i}{\partial \eta}=\sum_{k=1}^{n}\frac{\partial N^k}{\partial \eta}x_i+\sum_{k=1}^{n}\frac{\partial N^k}{\partial \eta}H^k V_{3i}^k \quad (i=1,2,3) \tag{19}$$

$$x_{i,\zeta}=\frac{\partial x_i}{\partial \zeta}=\sum_{k=1}^{n}N^k\frac{\partial H^k}{\partial \zeta}V_{3i}^k \quad (i=1,2,3) \tag{20}$$

$[\mathbf{R}]$ is defined as the inverse of the Jacobian matrix, $[\mathbf{J}]^{-1}$. Then the required partial derivatives are defined as:

$$\frac{\partial N^k}{\partial x_i}=R_{i1}\frac{\partial N^k}{\partial \xi}+R_{i2}\frac{\partial N^k}{\partial \eta} \quad (i=1,2,3) \tag{21}$$

$$\frac{\partial H^k}{\partial x_i}=R_{i3}\frac{\partial H^k}{\partial \zeta} \quad (i=1,2,3) \tag{22}$$

F. STRESS-STRAIN RELATIONSHIP

The strain calculated in Eq. (11) is in the global coordinate, and is transformed to a local coordinate system using

$$\{\varepsilon_{local}\}=[\mathbf{T}]\{\varepsilon_{\mathrm{global}}\} \tag{23}$$

where [*T*] is defined in Eq. (10). Stress is calculated from the strain in the local coordinate system using the plane-strain assumption for the in-plane stress components:

$$\begin{aligned} \sigma_x &= \frac{E}{1-\nu^2}(\varepsilon_x+\nu\,\varepsilon_y) & \tau_{xy} &= G\,\gamma_{xy} \\ \sigma_y &= \frac{E}{1-\nu^2}(\nu\,\varepsilon_x+\varepsilon_y) & \tau_{yz} &= \mathrm{K}\,G\,\gamma_{yz} \\ \sigma_z &= E\,\varepsilon_z & \tau_{xz} &= \mathrm{K}\,G\,\gamma_{xz} \end{aligned} \tag{24}$$

where K is the shear correction factor, *E* is the elastic modulus, *G* is the shear modulus, and ν is Poisson's ratio. The resulting stresses are in the local coordinate system and are converted to the global coordinate system with

$$\{\sigma_{global}\}=[\mathbf{T}]^{\mathrm{T}}\{\sigma_{local}\} \tag{25}$$

$$\{\sigma\}\equiv\{\sigma_{\mathrm{x}}\,\sigma_{\mathrm{y}}\,\sigma_{\mathrm{z}}\,\gamma_{\mathrm{xy}}\,\gamma_{\mathrm{yz}}\,\gamma_{\mathrm{xz}}\} \tag{26}$$

where [T] is from Eq. (10).

G. DRILLING MOMENTS

Let u_3 be the transverse deflection as defined in Eq. (6). The work done by a pressure loading, *p*, on an element is:

$$W=\int_{A^e} u_3\,p\,dA \tag{27}$$

where A^e is the element area. Substituting Eq. (6) into Eq. (27), the work is now:

$$W = \{u_3\}^T \int_{A^e} [\mathbf{N}] p \, dA + \{\theta_3\}^T \int_{A^e} [\mathbf{N}^k \mathbf{H}^k] p \, dA \tag{28}$$

The first term gives the conventional forces at each node, while the second term yields the new nodal load, which will be called the Drilling Moment (DM). For example, if p is a concentrated force P at node n, then the drilling moment associated with θ_3^n becomes $\frac{1}{2}\zeta \, tP$, where t is the shell thickness and the mid-plane is the reference plane.

When P is applied at the top plane, ζ equals one, and the drilling moment is $\frac{1}{2}tP$. If P is applied at the bottom plane (still with a positive loading direction), the drilling moment is $-\frac{1}{2}tP$. That is, the load results in compression or tension in the transverse normal stress depending on whether the loading is applied to the top or bottom surface of the shell. The transverse normal stress is assumed constant through the shell thickness for elastic deformation. However, as plastic deformation progresses, the transverse normal stress becomes non-uniform through the thickness in order to satisfy the yield function.

The drilling moment direction coincides with the direction of drilling rotation, but affects the transverse normal stress. The drilling moment is used as a mathematical convenience, and thus, lacks a physical interpretation.

H. INTERNAL FORCE, MASS AND THEIR ASSEMBLY

The stress resulting from Eq. (25) is then converted to an internal force vector and summed over all integration points using

$$\{f_{int}\}=\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}[\mathbf{B}]^{T}\{\sigma\}d\xi\, d\eta\, d\zeta=\sum_{i=1}^{nx}\sum_{j=1}^{ny}\sum_{k=1}^{nz}[\mathbf{B}]^{T}\{\sigma\}w_i\, w_j\, w_k|\mathbf{J}| \quad (29)$$

where | **J** | is the determinant of the Jacobian matrix, *nx*, *ny*, and *nz* are the number of integration points in the ξ, η, and ζ directions, respectively, and w_i, w_j, and w_k, are the Gauss weights related to those integration points. The resulting force vector, $\{f_{int}\}$, must have its components transformed back to the global coordinate system using:

$$\{F_{int}^{k}\}_{(6x1)}=\left|\begin{array}{cccccc}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&&&\\0&0&0&&[\mathbf{T}_p]&\\0&0&0&&&\end{array}\right|\{f_{int}^{k}\}_{(6x1)} \quad (k=1,2,3,4) \qquad (30)$$

A lumped mass method is used for the element mass matrix. The matrix for each element is diagonal, with equal diagonal elements:

$$[\mathbf{M}_e]=\begin{bmatrix}m_e&0&0&\dots&0\\0&m_e&0&\dots&0\\0&0&m_e&\dots&0\\\dots&\dots&\dots&\dots&\dots\\0&0&0&\dots&m_e\end{bmatrix} \qquad (31)$$

where

$$m_e = \frac{\left(\sum_{i=1}^{nx} \sum_{j=1}^{ny} \sum_{k=1}^{nz} w_i \, w_j \, w_k \left| \mathbf{J} \right| \right) \rho}{\mathrm{n}} \tag{32}$$

with $n = 4$ in this four-node element. Using a diagonal mass matrix greatly simplifies time integration, since inverting it is a trivial task requiring little computation time. The internal force vector and element mass vector (each 24 by 1) are then assembled into the corresponding system vectors.

I. EXPLICIT TIME INTEGRATION

The use of internal force vectors and explicit time integration negates the need to explicitly form the system stiffness matrices. The acceleration vector is computed from

$$\left\{ \ddot{U} \right\}^t = [\mathbf{M}]^{-1} \left(\left\{ F_{ext} \right\}^t - \left\{ F_{int} \right\}^t \right) \tag{33}$$

where $\left\{ \ddot{U} \right\}$ is the system acceleration vector, [**M**] is the system mass matrix, $\left\{ F_{ext} \right\}$ is the system external force vector, and superscript t denotes the time step. Of course, the system mass matrix in Eq. (33) is simply symbolic, since a system mass vector, formed from the diagonal of the mass matrix, is used in actual computation. Velocity and displacement are then found using

$$\begin{aligned} \left\{ \dot{U} \right\}^{t+\frac{\Delta t}{2}} &= \left\{ \dot{U} \right\}^{t-\frac{\Delta t}{2}} + \left\{ \ddot{U} \right\}^{t} \Delta t \\ \left\{ U \right\}^{t+\Delta t} &= \left\{ U \right\}^{t} + \left\{ \dot{U} \right\}^{t+\frac{\Delta t}{2}} \Delta t \end{aligned} \tag{34}$$

III. DAMAGE CONSTITUTIVE EQUATIONS

A. GURSON'S VOID MODEL

Yielding and plastic deformation in the element follows the model proposed by Gurson for symmetric deformations around a spherical void [17]. The yielding condition is

$$F = \left(\frac{q}{\sigma_0} \right)^2 + 2 q_1 \Phi \cosh\left(-\frac{3}{2} \frac{q_2 p}{\sigma_0} \right) - \left(1 + q_3 \Phi^2 \right) = 0 \qquad (35)$$

where Φ is the current porosity, p is the hydrostatic stress, q is the effective stress, and σ_0 is the current yield stress. This model assumes equivalent yield stress in both tension and compression. The constants q_1, q_2, and q_3 were introduced by Tvergaard in order to provide a better match with numerical studies [16]. Aravas provides a detailed explanation of implementing this model in a static finite element algorithm for three-dimensional solid elements [21]. The procedure used here is essentially the same, with some modifications due to the different element formulation. The stress is then transformed to local coordinates, and the internal force vector is computed as described above. One thing to be noted in Eq. (35) is that if Φ is initially 0, the yielding criteria surface is identical to the von Mises yield condition.

After calculating the strain tensor using Eq. (23), any previous plastic strain is subtracted using

$$\left\{ \varepsilon^e \right\} = \left\{ \varepsilon^{total} \right\} - \left\{ \varepsilon^p \right\} \qquad (36)$$

The stress is then calculated using Eq. (24) with the components of $\left\{\varepsilon^{e}\right\}$. Using these values, the hydrostatic stress, deviatoric stress and effective stress are calculated as follows:

$$p = -\tfrac{1}{3}\left(\sigma_{xx} + \sigma_{yy} + \sigma_{zz}\right) \tag{37}$$

$$\left\{s\right\} = \left\{\sigma\right\} + p\left\{\delta\right\} \tag{38}$$

$$q = \sqrt{\tfrac{3}{2}\left(s_1^2 + s_2^2 + s_3^2 + 2\left(s_4^2 + s_5^2 + s_6^2\right)\right)} \tag{39}$$

where δ is the Kronecker delta function:

$$\{\delta\} = \{11100\}^{T} \tag{40}$$

At this point, F is calculated from Eq. (35). If F is greater than zero, indicating plastic flow, iteration is required to determine the new porosity and change in plastic strain. The predictor-corrector method used in Aravas [21] is also used here. Using the values of p and q previously calculated as a first guess, correction factors are calculated using

$$\left\{C_f\right\} = [\mathbf{A}]^{-1}\left\{\mathrm{A1}\right\} \tag{41}$$

where

$$\left\{A1\right\} = \begin{Bmatrix} -F \\ -\Delta\varepsilon_p \frac{\partial F}{\partial q} - \Delta\varepsilon_q \frac{\partial F}{\partial p} \end{Bmatrix} \tag{42}$$

$$[\mathbf{A}] = \begin{bmatrix} \mathrm{K}\frac{\partial F}{\partial \mathrm{p}} + \frac{\partial F}{\partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_p} & -3\mathrm{G}\frac{\partial F}{\partial \mathrm{q}} + \frac{\partial F}{\partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_q} \\ \frac{\partial F}{\partial \mathrm{q}} + \Delta\varepsilon_{\mathrm{q}}\left[\mathrm{K}\frac{\partial^2 F}{\partial \mathrm{p}^2} + \frac{\partial^2 F}{\partial p \partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_p}\right] + \Delta\varepsilon_{\mathrm{p}}\frac{\partial^2 F}{\partial q \partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_p} & \frac{\partial F}{\partial \mathrm{p}} + \Delta\varepsilon_{\mathrm{p}}\left[-3\mathrm{G}\frac{\partial^2 F}{\partial \mathrm{q}^2} + \frac{\partial^2 F}{\partial q \partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_q}\right] + \Delta\varepsilon_{\mathrm{q}}\frac{\partial^2 F}{\partial q \partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_q} \end{bmatrix} \tag{43}$$

$$\left\{C_f\right\} = \begin{Bmatrix} \alpha \\ \beta \end{Bmatrix} \tag{44}$$

and

$$
\begin{aligned}
\tfrac{\partial F}{\partial q} &= \frac{2\,q}{\sigma_0^2} \\
\tfrac{\partial^2 F}{\partial q^2} &= \frac{2}{\sigma_0^2} \\
\tfrac{\partial F}{\partial p} &= 2\,q_1\,\Phi\,\left(\frac{-3\,q_2}{2\,\sigma_0}\right)\sinh\left(\frac{-3\,q_2\,p}{2\,\sigma_0}\right) \\
\tfrac{\partial^2 F}{\partial p^2} &= 2\,q_1\,\Phi\,\left(\frac{3\,q_2}{2\,\sigma_0}\right)^2\cosh\left(\frac{-3\,q_2\,p}{2\,\sigma_0}\right) \\
\tfrac{\partial F}{\partial \sigma_0} &= \frac{-q^2}{2\sigma_0^3} + 2q_1\Phi\frac{3q_2p}{2\sigma_0^2}\sinh\left(\frac{-3q_2p}{2\sigma_0^2}\right) \\
\tfrac{\partial^2 F}{\partial p\partial \sigma_0} &= -2q_1\Phi p\left(\frac{3q_2}{2\sigma_0^2}\right)^2\cosh\left(\frac{-3q_2p}{2\sigma_0^2}\right) + \frac{3q_2}{2\sigma_0^3}\sinh\left(\frac{-3q_2p}{2\sigma_0^2}\right) \\
\tfrac{\partial^2 F}{\partial q\partial \sigma_0} &= \frac{-q}{\sigma_0^3}
\end{aligned}
\tag{45}
$$

The values for α and β are used to correct the change in strain caused by hydrostatic pressure and the change in strain caused by effective stress (See Eq. (46)), which are then used to determine the change in the plastic strain vector, as shown in Eq. (47).

$$
\begin{aligned}
\Delta\varepsilon_p^{new} &= \Delta\varepsilon_p^{old} + \alpha \quad (\Delta\varepsilon_p^0 = 0) \\
\Delta\varepsilon_q^{new} &= \Delta\varepsilon_q^{old} + \beta \quad (\Delta\varepsilon_q^0 = 0)
\end{aligned}
\tag{46}
$$

$$
\{\Delta\varepsilon^p\} = \tfrac{1}{3}\Delta\varepsilon_p\{\delta\} + \Delta\varepsilon_q\left(\frac{3}{2\,q}\right)\{s\}
\tag{47}
$$

This change in plastic strain, $\{\Delta\varepsilon^p\}$, is then added to the total plastic strain, and the new elastic strain is calculated using Eq. (36), and the stress vector is calculated again using Eq. (24). Next, the change in void content, or porosity, is calculated as:

$$
\Delta\Phi = \Delta\Phi_{growth} + \Delta\Phi_{nucleation}
\tag{48}
$$

$$
\Delta\Phi_{growth} = (1-\Phi)\sum\varepsilon_{ii}^p
\tag{49}
$$

$$\Delta \Phi_{nucleation} = \frac{\Phi_N}{s_N \sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{\varepsilon^p - \varepsilon_N}{s_N} \right)^2 \right] \Delta \varepsilon_p \tag{50}$$

where Φ_N is the volume fraction of void nucleating particles and ε_N and s_N are the mean and standard deviation of a normal distribution of nucleation strain, as suggested by Chu and Needleman [15] and utilized by Aravas [21]. Then the effective plastic strain, and void content are updated with

$$\Delta \varepsilon^p = \frac{-p\,\Delta \varepsilon_p + q\,\Delta \varepsilon_q}{(1 - \Phi\ )\sigma_0} \tag{51}$$

$$\varepsilon^p_{t+\Delta t} = \varepsilon^p_t + \Delta\varepsilon^p \tag{52}$$

where ε^p_t is the effective plastic strain from the previous time step. At this point, the change in yield stress due to strain hardening is calculated (see below). If either α or β is greater than a predetermined tolerance, the process iterates beginning with Eq. (41). The tolerance used for all examples presented in the paper is 1.0 x 10^{-4}.

B. IMPROVING STABILITY IN THE CONSTITUTIVE EQUATIONS

Stability in the procedure outlined above is very dependent on the order in which the various equations are evaluated. Aravas discusses the stability problem when considering problems involving large plastic strains [21]. While the change in plastic strain from one time step to the next will theoretically be small, high strain rate and changes in the strain-hardening characteristics of the material act to degrade stability. Equations (42), (43) and (45) are not complete in that they do not contain all of the derivative terms required by the chain rule.

The predictor-corrector method used is based on Newton's method, where the correction factors, α and β, are found from

$$\begin{bmatrix} f_{1,\Delta\varepsilon_p} & f_{1,\Delta\varepsilon_q} \\ f_{2,\Delta\varepsilon_p} & f_{2,\Delta\varepsilon_q} \end{bmatrix} \begin{Bmatrix} \alpha \\ \beta \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \tag{53}$$

where

$$f_1 = \left(\frac{q}{\sigma_0}\right)^2 + 2q_1\Phi\cosh\left(\frac{-3q_2 p}{2\sigma_0}\right) - \left(1 + q_3\Phi^2\right) \tag{54}$$

$$f_2 = \Delta\varepsilon_p \frac{\partial f_1}{\partial q} + \Delta\varepsilon_q \frac{\partial f_1}{\partial p} \tag{55}$$

$$f_{1,\Delta\varepsilon_p} = K\frac{\partial f_1}{\partial p} + \frac{\partial f_1}{\partial \Phi}\frac{\partial \Phi}{\partial \Delta\varepsilon_p} + \frac{\partial f_1}{\partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_p} \tag{56}$$

$$f_{1,\Delta\varepsilon_q} = -3G\frac{\partial f_1}{\partial q} + \frac{\partial f_1}{\partial \Phi}\frac{\partial \Phi}{\partial \Delta\varepsilon_q} + \frac{\partial f_1}{\partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_q} \tag{57}$$

$$f_{2,\Delta\varepsilon_p} = \frac{\partial f_1}{\partial q} + \Delta\varepsilon_p \frac{\partial^2 f_1}{\partial \sigma_0 \partial q}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_p} + \Delta\varepsilon_q\left[K\frac{\partial^2 f_1}{\partial p^2} + \frac{\partial^2 f_1}{\partial p \partial \Phi}\frac{\partial \Phi}{\partial \Delta\varepsilon_p} + \frac{\partial^2 f_1}{\partial p \partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_p}\right] \tag{58}$$

$$f_{2,\Delta\varepsilon_q} = \frac{\partial f_1}{\partial p} + \Delta\varepsilon_q\left[\frac{\partial^2 f_1}{\partial p \partial \Phi}\frac{\partial \Phi}{\partial \Delta\varepsilon_q} + \frac{\partial^2 f_1}{\partial p \partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_q}\right] + \Delta\varepsilon_p\left[-3G\frac{\partial^2 f_1}{\partial q^2} + \frac{\partial^2 f_1}{\partial q \partial \sigma_0}\frac{\partial \sigma_0}{\partial \Delta\varepsilon_q}\right] \tag{59}$$

Derivative terms that are zero have been dropped, and *K* and *G* are the bulk and shear moduli, respectively. f_1 is Gurson's void function, and f_2 is the flow rule, both of which are driven to zero. As void content increases, the number of iterations required to achieve convergence increases dramatically, and often diverges, instead. By removing all partial derivatives relating to void content, the stability of the algorithm is greatly increased, at the cost of only 2 to 3 extra iterations, depending on the current void content. The dependence on the current yield stress is retained to allow convergence across a transition in the slope of the yield stress versus strain relationship.

The partial derivatives of yield stress with respect to $\Delta\varepsilon_p$ and $\Delta\varepsilon_q$ are computed by calculating the slope from the current yield stress to the yield stress corresponding to the increase in plastic strain from the corrected values of $\Delta\varepsilon_p$ and $\Delta\varepsilon_q$. This means that these derivatives will be zero on the first iteration, since both of the control variables are initialized to zero. Several error traps must be included to prevent round-off and truncation error from causing the algorithm to diverge, and to prevent porosity from decreasing or becoming negative. One of the assumptions used in this implementation is that once voids form, they do not disappear. In other words, voids do not disappear when the element is placed in compression.

C. STRAIN HARDENING

Modeling the nonlinear elasto-plastic behavior of the material used is simplified by constructing a piece-wise linear version of the stress-strain plot. Using the tangent modulus, E_T, for each piece-wise region, the yield stress is calculated with

$$\sigma_0^{t+\Delta t} = \sigma_0^0 + \sum_{i=1}^{j-1} E_{Ti}\left(\varepsilon_i - \varepsilon_{i-1}\right) + E_{Tj}\left(\varepsilon_t^{eff} - \varepsilon_{j-1}\right) \qquad (60)$$

where $\sigma_0^{t+\Delta t}$ is the yield stress for this time step, σ_0^0 is the original yield stress, ε_t^{eff} is the total effective strain for the time step under consideration, and ε_i is the upper strain limit of the *i*th linear segment. ε_0 is the original yield strain, and is calculated by the program as simply σ_0^0/E. The current effective elastic strain is calculated by dividing the current yield stress by the elastic modulus, *E*. This is added to the current effective plastic strain to obtain the total effective strain.

As an example, let the total effective strain from Eq. (49) lie within the second work-hardening segment. The new yield stress is

$$\sigma_0^{t+\Delta t} = \sigma_0^0 + E_{T1}\left(\varepsilon_1 - \varepsilon_0\right) + E_{T2}\left(\varepsilon_{t+\Delta t}^{eff} - \varepsilon_1\right) \tag{61}$$

Figure 3 illustrates this example. The algorithm calculates the new yield stress as a function of the cumulative effective plastic strain, the current effective elastic strain, and the original yield stress for each iteration of the damage constitutive equations.

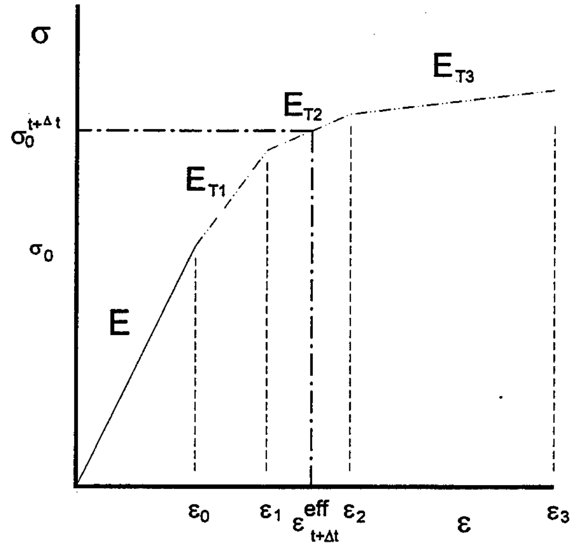


Figure 3. Calculating a New Yield Stress.

IV. HOURGLASS MODE CONTROL

Only one integration point is used in each plane parallel to the reference plane, which results in under-integration in the ξ-η plane. This leads to spurious, hourglass, or zero-energy modes in the element, which will yield useless results if left uncontrolled. The effects of these modes are shown in Fig. 4, a pinched cylinder where one eighth of the structure is modeled by utilizing symmetry boundary conditions. Clearly, no useful information can be obtained from the resulting solution.

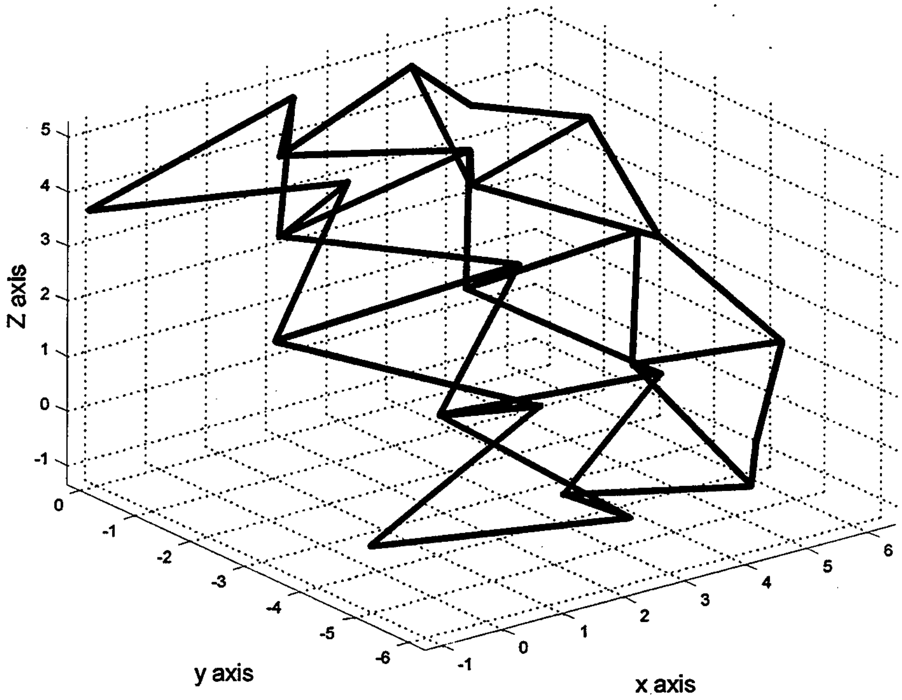


Figure 4. Hourglass Modes in a Pinched Cylinder Model.

Belytschko, et al., proposed an efficient means of controlling the hourglass modes of a similar element [25]. The method described uses a portion of a stiffness matrix generated by full integration in all directions to modify the stiffness matrix generated by under-integration. Although the formulation proposed in this paper does not use a stiffness matrix, a similar approach is just as effective in controlling these modes.

Rather than fully integrating in all three directions, the element is fully integrated in the ξ-η plane, but under-integrated in the ζ direction, as shown in Fig. 5. The procedure described above for calculating the internal force vector is followed to generate an internal force vector related to these four integration points.

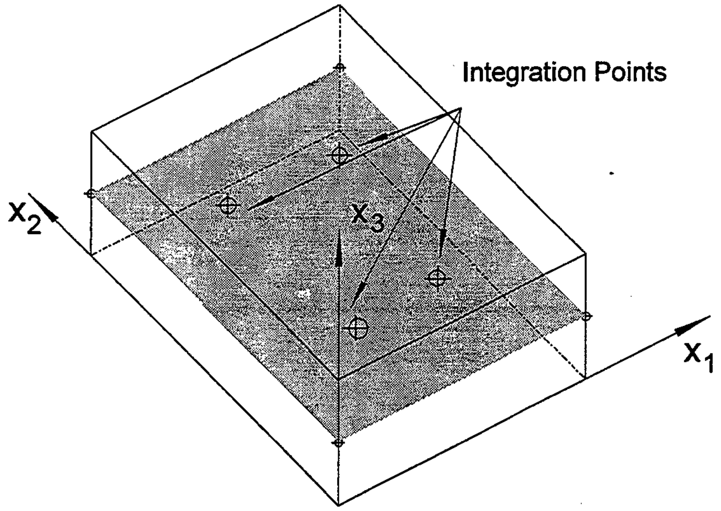


Figure 5. Hourglass Mode Control Integration Points.

The algorithm for calculating strain, stress, and force for the hourglass mode control integration points is identical to the algorithm used to produce the main internal force vector, with the exception of plastic strain. The damage constitutive equations described in the previous section are not utilized for hourglass mode control. The internal forces generated from the two integration schemes are treated like the stiffness matrices in Belytschko et al. [25]. For *nz* integration points through the element thickness, the new force vector is

$$\left\{ f_{int} \right\} = \left\{ f_{int}^{1x1xnz} \right\} + h \left\{ f_{hour} \right\} \tag{62}$$

where

$$\left\{ f_{hour} \right\} = \left\{ f_{int}^{2x2x1} \right\} - \left\{ f_{int}^{1x1xnz} \right\} \tag{63}$$

and

$$h = \frac{r\, t^2}{A} \tag{64}$$

The variable *h* is used here in Eqs. (62) and (64) instead of the ε used in Belytschko et al. [25], to avoid confusion with the various strains discussed in this paper. The variables used to calculate *h* are the element thickness (*t*) and the element's surface area (*A*). The effect of *r* follows that described in Belytschko et al. [25], and is set to 0.05. The range of values for *r* that effectively controls the hourglass modes, but does not greatly affect the overall element stiffness is roughly 0.046 to 0.057 (determined experimentally). Since the elements are in arbitrary orientation in 3-D space, the area calculation is computed as the sum of the area of the two triangles formed by dividing the element at the diagonal between nodes one and three:

$$A = \tfrac{1}{2} \sqrt{l_1^2 l_2^2 - D_1^2 + l_3^2 l_4^2 - D_2^2} \tag{65}$$

where

$$D_1 = (x_1^1 - x_1^2)(x_1^3 - x_1^2) + (x_2^1 - x_2^2)(x_2^3 - x_2^2) + (x_3^1 - x_3^2)(x_3^3 - x_3^2)$$
$$D_2 = (x_1^1 - x_1^4)(x_1^3 - x_1^4) + (x_2^1 - x_2^4)(x_2^3 - x_2^4) + (x_3^1 - x_3^4)(x_3^3 - x_3^4) \qquad (66)$$

and

$$l_1 = \sqrt{\left(x_1^2 - x_1^1\right)^2 + \left(x_2^2 - x_2^1\right)^2 + \left(x_3^2 - x_3^1\right)^2}$$
$$l_2 = \sqrt{\left(x_1^3 - x_1^2\right)^2 + \left(x_2^3 - x_2^2\right)^2 + \left(x_3^3 - x_3^2\right)^2}$$
$$l_3 = \sqrt{\left(x_1^4 - x_1^3\right)^2 + \left(x_2^4 - x_2^3\right)^2 + \left(x_3^4 - x_3^3\right)^2} \qquad (67)$$
$$l_4 = \sqrt{\left(x_1^1 - x_1^4\right)^2 + \left(x_2^1 - x_2^4\right)^2 + \left(x_3^1 - x_3^4\right)^2}$$

and $\left(x_1^k, x_2^k, x_3^k\right)$ is the location of node k in the global coordinate system. For these calculations, the element is assumed to be flat (no curvature along either the ξ or η directions). The effectiveness of this method of control is shown in Fig. 6, the same problem as shown in Fig. 4, but with the hourglass mode control described above.

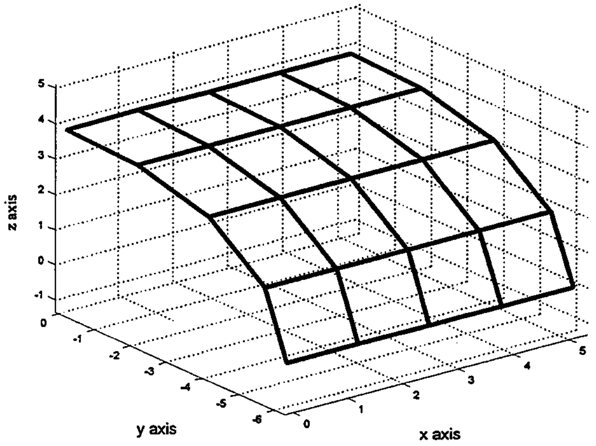


Figure 6. Pinched Cylinder Model with Hourglass Mode Control.

All verification problems analyzed in the following section were completed with hourglass control enabled. The current implementation uses hourglass control consistently, but allows easy modification to make hourglass control a user-defined option. The internal hourglass mode control can be disabled when the element is used in a general finite element program that includes various methods of spurious mode control.

V. IMPLEMENTATION IN RESEARCH-ORIENTED FE CODE

An in-house general purpose finite element analysis program, referred to herein as FEA, was used to verify the algorithms discussed. Several of these verification problems are presented in the next section. The program is written in FORTRAN, and is tailored for easy modification, rather than speed or ability to handle very large problems. The program reads the system parameters from a single input file, then writes the results to two ASCII text files: one contains the nodal coordinates and displacements for all time steps, the other contains the data associated with each element for all time steps (stress and strain tensors, effective plastic strain, void content, yield stress, etc.).

A. PREPROCESSOR

Since this program is locally generated, there is no preprocessor or postprocessor available. In order to facilitate running several problems, a preprocessor was written in MATLAB that generates the proper input file for several simple geometric shapes. The MATLAB script file will create a properly meshed flat plate, open box, cylinder, complete spherical cap, or a spherical cap with a hole in its top. The listing for the preprocessor is presented in Appendix A.

B. POSTPROCESSOR

A postprocessor, also written as a MATLAB script file, utilizes a graphical user interface (GUI) to create the required graphs. It also has the capability to create an animated display of the displacement over time. The displacements are scaled to make them clearly visible. This feature is useful in determining the effectiveness of hourglass control and verifying the boundary conditions applied. However, the MATLAB movie is memory intensive and slow, and is therefore useful only as a qualitative verification of proper problem definition. All graphs of problem solutions, except as noted, were generated using this postprocessor. The full listing of all associated files is provided in Appendix B.

C. FEA SHELL ELEMENT SUBROUTINE DESCRIPTION

The FORTRAN source code for the FEA implementation of the element presented here, along with the proposed hourglass control and constitutive model, is listed without comment in Appendix C. Since the primary purpose of the FEA program is to assist in the development of finite element algorithms, this subroutine is written with readability, rather than computational efficiency, as a primary concern,. A brief outline of the subroutine is shown in Fig. 7. The FEA program treats each type of element (shell, beam, etc.) in a single block, and loops through each element, then each integration point within the element.

The order of calculation when evaluating the constitutive equations is critical, and any variation will cause instability. The two control variables, *dep* and *deq*, are updated by the correction factors on each iteration. All other variables are initialized to the values from the previous time step within each iteration. This essential feature allows stability of the algorithm. As discussed above, the variation of void content with *dep* and *deq* is not used, but the variation of yield stress with the two control variables is calculated and included in the full expansion of the partial derivatives.

Program Outline for ELSH4L.F

```
1.  Enter subroutine
2.  Set constants
3.  Load gauss points and weights
4.  Begin loop over each shell element
    A.  Initialize element internal force vectors
    B.  Retrieve element material properties
    C.  Compute element indices into system vectors
    D.  Load nodal coordinates and displacements
    E.  Compute local coordinate system and surface area
    F.  Compute rotation matrices
    G.  Transform displacements to local coordinate system
    H.  Calculate hourglass force vector
    I.  Begin Integration Loop
        1)  Load shape functions and derivatives
        2)  Calculate Jacobian, its inverse, and its determinate
        3)  Calculate strain-displacement matrix ([B])
        4)  Compute strain ($\{\varepsilon\} = [B]\{d\}$)
        5)  Transform strain to local coordinate system
        6)  Subtract any previous plastic strain from strain tensor
        7)  Load last values of $\sigma_0$, $\phi$, and $\varepsilon^p_{eff}$
        8)  Compute stress ($\{\sigma\} = [D]\{\varepsilon\}$)
        9)  Compute yield function (F)
        10) If $F > 0$, set dep = deq = 0 and begin iteration
            a)  Calculate required partial derivatives
            b)  Prevent Numerical Errors (Divide by Zero, etc.)
            c)  Retrieve Previous $\phi$, $\varepsilon^p_{eff}$, and $\sigma_0$
            d)  Update dep and deq
            e)  Compute Incremental Plastic Strain Tensor
            f)  Compute Void Growth
            g)  Compute New $\sigma_{ij}$, $s_{ij}$, and $\varepsilon^p_{eff}$
            h)  Compute Void Nucleation Factor
            i)  Compute New $\phi$ and $\sigma_0$
            j)  Recalculate Yield Function (F)
            h)  If F is not Zero, go to step a)
        11) Last values used are retained.
        12) Transform stress to global coordinate system
        13) Calculate internal force for this integration point and sum
        14) Store material tensors and constants in system vectors
        15) End of integration loop
    J.  Apply hourglass correction force
    K.  Transform internal force vector to global coordinate system
    L.  Compute element mass vector
    M.  Load mass and force vectors into system vector
    N.  End of element loop
5.  End of subroutine
```

Figure 7. Outline of FEA Implementation.

VI. NUMERICAL EXAMPLES

The element presented here has been extensively tested using a variety of problems in which an analytic solution was available, and has produced satisfactory results in all cases studied. The transformation matrices and constitutive equations were verified with specifically tailored problems, and the element passes the patch test. The examples presented below are representative of the test cases used to verify the element, and are presented in the order of curvature: flat, singly curved, and doubly curved. For each case, an elastic solution is compared to available results, and then the elasto-plastic solutions are presented.

A. ELASTIC PLATE

A plate clamped on all four sides is subjected to a concentrated force at its center. The elastic modulus is 10 Msi (68.95 GPa), the density is 0.1647 slugs/in^3 (0.147 kg/cm^3), and Poisson's ratio is 0.2. The dimensions of the plate are 10 in x 10 in x 0.1 in thick (25.4cm x 25.4 cm x 0.254 cm). The yield stress is set high enough to ensure a completely elastic response. Two integration points through the thickness are used. The applied force is 40 lbf (177.9 N). The finite element mesh uses symmetry to model one quarter of the plate, with appropriate symmetry boundary conditions applied. Both a 2x2 (4 element) and 4x4 (16 element) mesh are used in the finite element analysis. The results for both meshes and the analytic solution are shown in Table 1. Using a four-element mesh, the new shell element obtained a displacement within 3.27% of the analytic solution, and 0.82% using a 16 element mesh.

Table 1. Comparison of Results for Elastic Clamped Plate.

Analysis Type	Center Node Peak Displacement (in)
2 x 2 FE Mesh (Dynamic)	-4.74×10^{-2}
4 x 4 FE Mesh (Dynamic)	-4.94×10^{-2}
Analytic (Twice the Static Solution)	-4.90×10^{-2}

B. THICK CLAMPED PLATE UNDER PRESSURE LOAD IN ELASTO-PLASTIC REGION

A thick steel plate, clamped on all four sides, is subjected to dynamic pressure load. The plate is 6m by 6m by 0.6m thick (for a 10:1 ratio). Table 2 shows the material properties for the structure. Table 3 shows the properties for the void model.

Table 2. Material Properties of Clamped Plate.

Property	Value	Units
Elastic Modulus (E)	2×10^{11}	Pa
Tangent Modulus (E_T)	2×10^{10}	Pa
Density (ρ)	7850	kg/m^3
Poisson's ratio (ν)	0.29	(none)
Yield Stress (S_{YP})	2.5×10^{8}	Pa

Table 3. Void Characteristics of Clamped Plate.

Initial Void Content (Φ_0)	0.0
Nucleating Particle Content (Φ_N)	0.04
Mean Nucleation Strain (e_N)	0.3
Nucleation Strain Standard Deviation (s_N)	0.1
Model Constant q_1	1.5
Model Constant q_2	1.0
Model Constant q_3 (= q_1^2)	2.25

One quarter of the plate is modeled with a three by three element mesh, for a total of nine elements, with appropriate symmetry boundary conditions applied, and is shown below in Fig. 8.

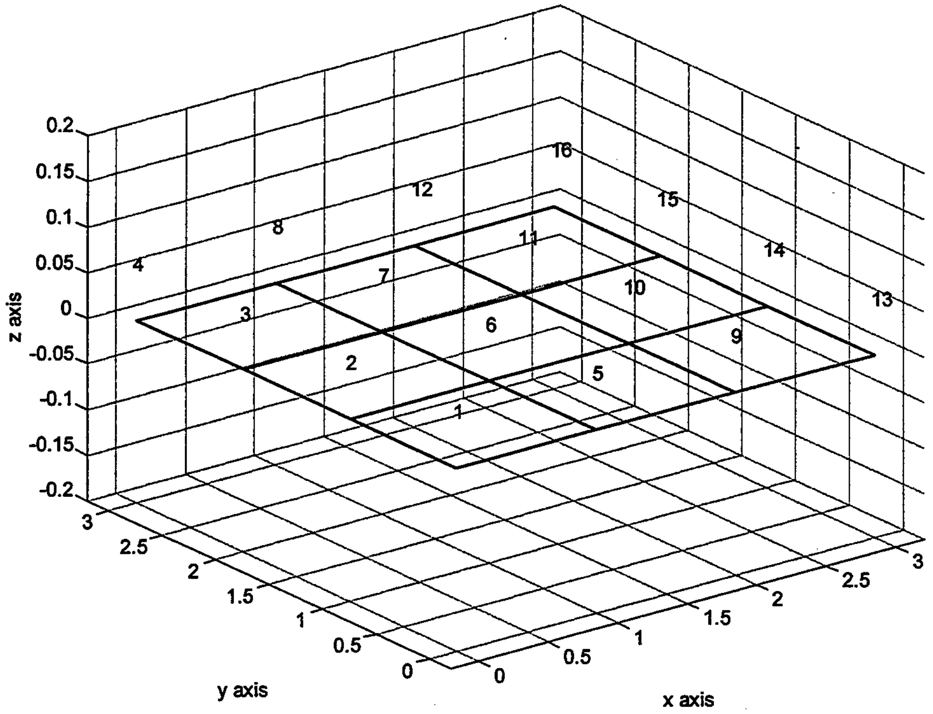


Figure 8. Nine Element Clamped Plate Mesh.

Four integration points are used through the thickness of each element. The calculation time step is 10^{-5} seconds, while the data is plotted at $2x10^{-4}$ second increments. The analysis terminates at 0.04 seconds. The plate is subjected to a uniformly distributed pressure acting downwards, and includes the appropriate drilling moment. The pressure increases linearly from 0.0 Pa at the start to 80 MPa at 0.01 seconds, then remains constant for the duration of the analysis. Three cases were analyzed: no void effects, void growth effects only, and void growth and nucleation. Figure 9 illustrates the effect of voids on the effective stress versus the effective strain relationship in the top of one of the border elements. When void effects are not included, the Von-Mises Equivalent (VME) stress follows the yield stress in the plastic region (See Fig. 9a).

Including void effects causes the yielding before the VME stress reaches the yield stress. As the void content increases with continued plastic flow, the VME stress falls farther below the yield stress, as shown in Fig. 9b (see Eq. (35)). Table 4 summarizes the results of the three cases. The most significant difference is in the transverse normal stress, σ_{zz}. This difference resulted in a 2.0% increase in the effective plastic strain, and a 2.6% increase in the peak deflection of the center of the plate.

Another interesting point is that the transverse normal stress was compressive and constant throughout the thickness up until plastic flow, as assumed in the formulation, then varied as the stress in the bottom fiber decreased and became tensile. Figure 10 shows the variation of transverse normal stress through the thickness of the element at the time of peak stress. Eventually, the transverse normal stress varied from compressive at the top, where the pressure was applied, and tensile at the bottom. The examples that follow will not include a separate analysis for void growth effects only, since the difference of including nucleation effects at the resulting small plastic strains obtained do not cause significantly different results.

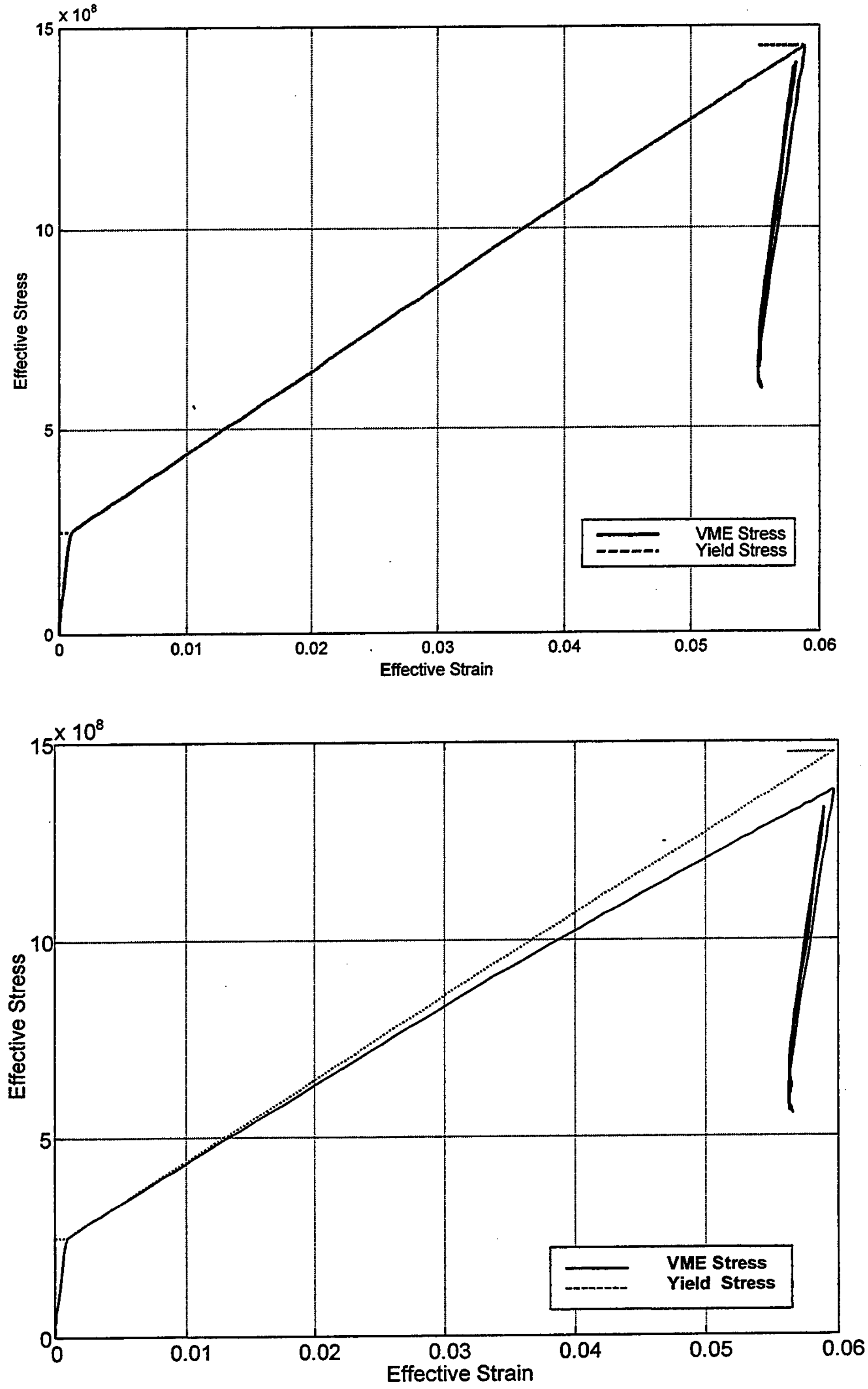


Figure 9. Void Effects in a Clamped Plate with Pressure Loading: Top (a) - No Void Effects, Bottom (b) - Void Effects.

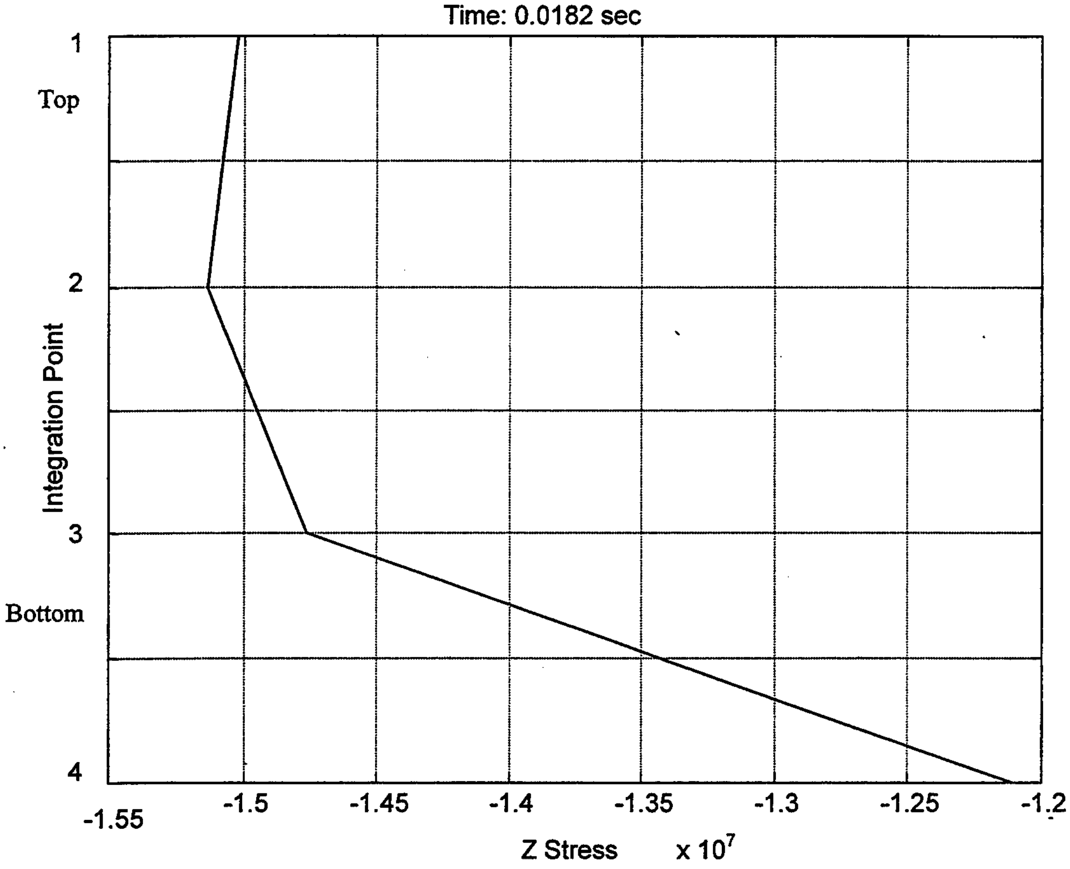


Figure 10. Transverse Normal Stress Variation through Shell Thickness: Clamped Plate with Pressure Loading and Void Effects.

Table 4. Summary of Results for Clamped Plate with Pressure Load.

Peak Values for Element #3	No Void Effects	Void Growth	Growth and Nucleation
σ_{VM} (Gpa)	1.4498	1.3789	1.3784
$\varepsilon_{effective}$	0.0588	0.0597	0.0597
σ_{xx} (GPa)	1.4501	1.3603	1.3598
σ_{yy} (GPa)	0.6561	0.6030	0.6073
σ_{zz} (GPa) (Max Tension)	-0.0003	0.0065	0.0091
σ_{zz} (GPa) (Max Compression)	-0.0363	-0.0322	-0.0320
$\varepsilon^{plastic}$ (effective)	0.0540	0.0551	0.0551
Φ (Porosity)	NA	0.0355	0.0357
Deflection (m) (Center Node)	-0.3801	-0.3897	-0.3898

C. THICK CLAMPED PLATE WITH CENTRAL POINT LOAD IN THE ELASTO-PLASTIC REGION

The geometry and material properties of this example are identical to those used in the previous example. The pressure load has been replaced with a point load at the center node. Four cases are studied: without void effects or a drilling moment, with void effects, with a drilling moment applied, and with both void effects and a drilling moment applied. All four sides are clamped, and the center node displacement is restricted in the x and y directions. Then, a DM equal to the applied force times one-half the plate thickness is applied for both the no-void and void cases. The time history of the stress components in the center element is shown in Fig. 11, and the relationship between porosity and effective plastic strain is illustrated in Fig. 12. The results for all six cases are summarized below in Table 5.

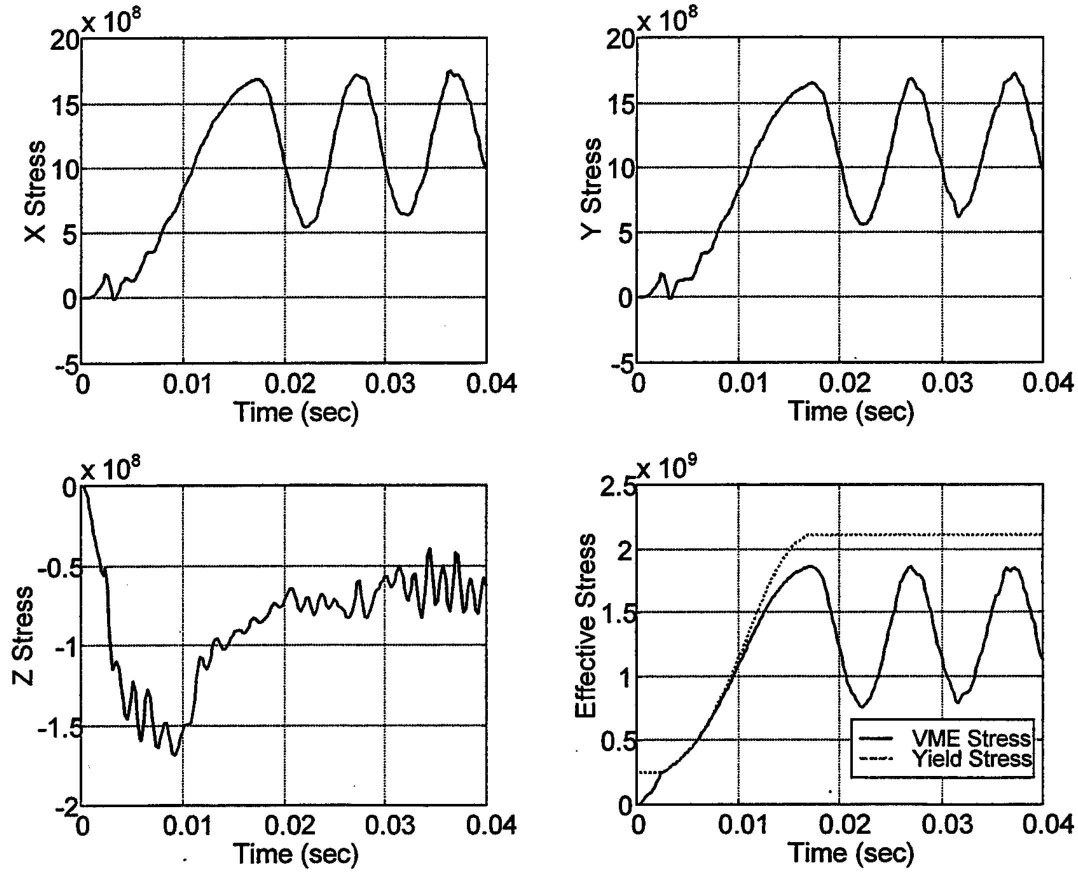


Figure 11. Stress Component Time History in Bottom Fiber: Clamped Plate with Concentrated Load, Void Effects, and Drilling Moment Applied.

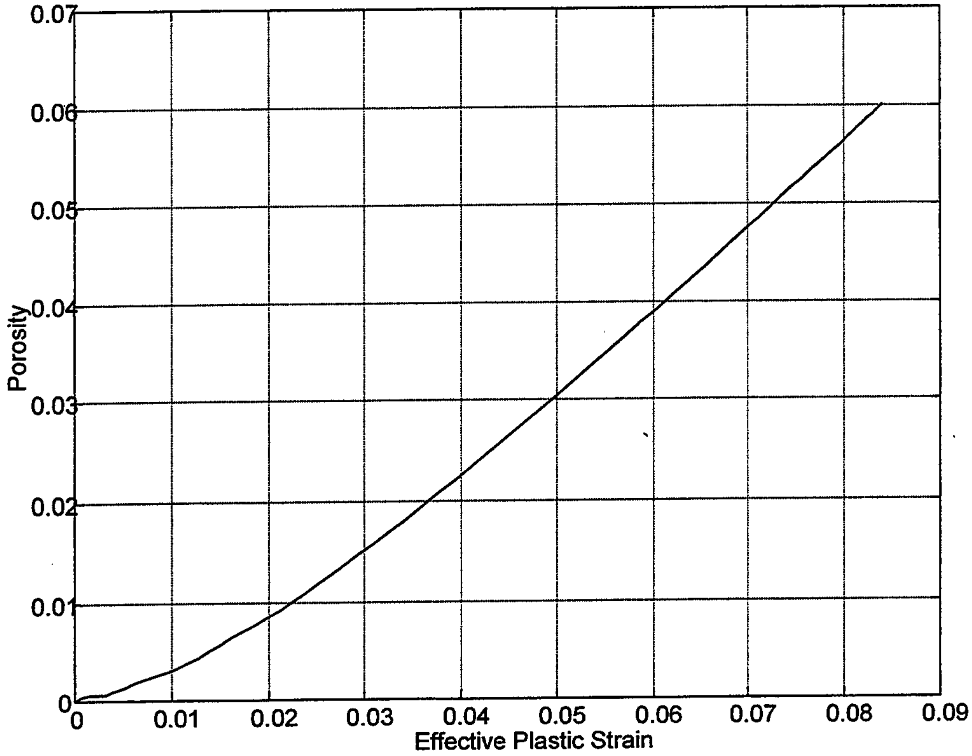


Figure 12. Porosity versus Effective Plastic Strain in Bottom Fiber: Clamped Plate with Concentrated Load, Void Effects, and Drilling Moment Applied.

Table 5. Summary of Results for Clamped Plate Subjected to Point Force.

Peak Values for Center Element	No Voids No DM	Voids No DM	No Voids Drilling Moment	Voids Drilling Moment
σ_{VM} (GPa)	2.1119	1.9021	2.0540	1.8590
$\varepsilon_{effective}$	0.0892	0.0929	0.0865	0.0887
σ_{xx} (GPa)	2.0008	1.8407	2.0000	1.7498
σ_{yy} (GPa)	1.9588	1.7245	1.8589	1.7234
σ_{zz} (GPa) (Max Tension)	0.0048	0.0213	-0.0003	-0.0003
σ_{zz} (GPa) (Max Compression)	-0.0036	-0.0057	-0.1751	-0.1691
$\varepsilon^{plastic}$ (effective)	0.0838	0.0880	0.0812	0.0840
Φ (Porosity)	NA	0.0663	NA	0.0601
Deflection (m) (Center Node)	-0.4413	-0.4544	-0.4350	-0.4451

The drilling moment increases the effective stress on the fiber in compression, and decreases the effective stress on the fiber in tension. The reduction in the tensile stress results in a reduction in the effective plastic strain. Including void effects decreases the VME stress on the fiber in tension, and slightly increases the VME stress on the fiber in compression. The effective plastic strain is also increased. All of these results indicate that this formulation correctly predicts, in a qualitative sense, the effects of drilling moments and void growth and nucleation. Including only void effects increased the effective plastic strain by 5.0%, while including only the drilling moment decreased the effective plastic strain by 3.1%. Including both void effects and the drilling moment increased the effective plastic strain by 0.2%, indicating the importance of applying both effects together.

D. SIMPLY SUPPORTED PLATE WITH CENTRAL POINT LOAD IN THE ELASTO-PLASTIC REGION

The material properties, geometry, mesh, and time values from the clamped plate problem above are used here. The magnitude of the force is 8x10^8 N, and the drilling moment, when applied, is –2.4x10^8 N. The same four cases are analyzed: no void or drilling moment effects, drilling moment effects only, void effects only, and both void and drilling moment effects. The analysis results for all four cases are summarized in Table 6.

Table 6. Summary of Results for a Simply Supported Plate with a Point Force.

Peak Values for Center Element	No Voids No DM	No Voids Drilling Moment	Voids No DM	Voids and Drilling Moment
σ_{VM} (GPa)	3.0874	3.0089	2.5061	2.4881
$\varepsilon_{effective}$	0.1360	0.1321	0.1450	0.1393
σ_{xx} (GPa)	3.0253	2.9123	2.4686	2.4206
σ_{yy} (GPa)	3.0459	2.8889	2.4612	2.3928
σ_{zz} (GPa) (Max Tension)	0.0060	-0.0003	0.0114	-0.0003
σ_{zz} (GPa) (Max Compression)	-0.0035	-0.1802	-0.0035	-0.1722
$\varepsilon^{plastic}$ (effective)	0.1277	0.1242	0.1382	0.1327
Φ (Porosity)	NA	NA	0.1197	0.1092
Deflection (m) (Center Node)	-0.9219	-0.8990	-0.9770	-0.9420

The qualitative results for this example are the same as for the clamped plate. Since the applied force causes greater initial yielding, void growth and nucleation has a greater effect. The drilling moment, when added to the void effects, reduces the amount of effective plastic strain in tension and displacement.

E. ELASTIC PINCHED CYLINDER

An open-ended cylinder of radius 5.0 in. (12.7 cm), length 10.35 in. (26.289 cm), and thickness 0.094 in. (0.23876 cm) is subjected to a pinching load of 100 lbf (444.82 N) (See Fig. 6). The elastic modulus is 10.5 msi (72.4 Gpa), Poisson's ratio is 0.3125, and the density is 3.125x10^{-3} slugs/in^3 (2783 kg/m^3). The load is applied as a step function beginning at time t = 0 seconds. Using symmetry, the problem was reduced to a one-eighth section of the cylinder. The dynamic value should be twice the analytic static value. Inextensional shell theory gives a static radial contraction of 0.1117 in. (0.28372 cm). The maximum radial contraction of the model is 0.1995 in. (0.5067 cm), which translates to a static radial contraction of 0.09975 in (0.2534 cm). Using a 256-element mesh (16 by 16), the maximum radial contraction was 0.2207 in. (0.5606 cm), for a static contraction of 0.1104 in (0.2804 cm). The results are summarized in Table 7. The 16-element solution is within 10.7% of the analytic solution, while the 256-element mesh is within 1.21%.

Table 7. Comparison of Results for Elastic Pinched Cylinder.

Analysis Type	Radial Contraction (in)
Finite Element with 16 Element Mesh	0.1995 (0.5067 cm)
Finite Element with 256 Element Mesh	0.2207 (0.5606 cm)
Analytic (Twice the Static Solution)	0.2234 (0.5674 cm)

F. THICK PINCHED CYLINDER IN THE ELASTO-PLASTIC REGION

The material and void properties shown in Tables 3 and 4 are used for an open-ended cylinder with a pinching force at its center. The top half of the cylinder is modeled using a 36 element mesh with appropriate symmetry boundary conditions along the bottom edge of mesh, as shown in Fig. 13. The analysis is calculated in 10^{-5} second steps, with output every $2\text{x}10^{-4}$ seconds, from 0.0 seconds to 0.04 seconds. The pinching force is 66.792 kN on each side, with a drilling moment of 848.258 Nm applied on the cases indicated. The same four cases are compared: no voids or drilling moment, void effects only, drilling moment only, and both void effects and drilling moment (DM). The effects of void growth and nucleation and the drilling moment is shown in the stress component time histories; Fig. 14 shows the stresses without voids or a drilling moment, and Fig. 15 shows the stresses with both effects included. The results for all cases are summarized in Table 8.

Table 8. Summary of Results for Elasto-Plastic Pinched Cylinder.

Peak Values for Center Element	No Voids No DM	No Voids Drilling Moment	Voids No DM	Voids and Drilling Moment
σ_{VM} (MPa)	262.40	265.05	259.28	283.49
$\varepsilon_{effective}$ (x 10^3)	1.6504	1.5722	1.4273	2.7049
σ_{xx} (MPa)	182.46	160.38	170.12	150.31
σ_{yy} (MPa)	291.71	283.77	291.83	297.97
σ_{zz} (MPa) (Max Tension)	5.0619	-0.0238	1.0270	7.5235
σ_{zz} (MPa) (Max Compression)	-7.6358	-28.739	-7.0364	-35.373
$\varepsilon^{plastic}$ (effective) (x 10^3)	0.7828	0.7583	0.5771	1.6923
Φ (Porosity) (x 10^3)	NA	NA	0.4773	1.1970
Deflection (mm) (Center Node) Global Maximum	2.7589	12.919	-0.0220	-0.0214
Deflection (mm) (Center Node) Global Minimum	-31.750	-31.750	-31.750	-31.750

Since there is only a small amount of plastic strain, the effects of void growth and nucleation are greatly reduced. Even with this small amount of plastic flow, the effective plastic strain was increased by over 116 percent with the inclusion of both drilling moment and void effects, while the peak center node displacement was only increased by approximately 4.5 percent.

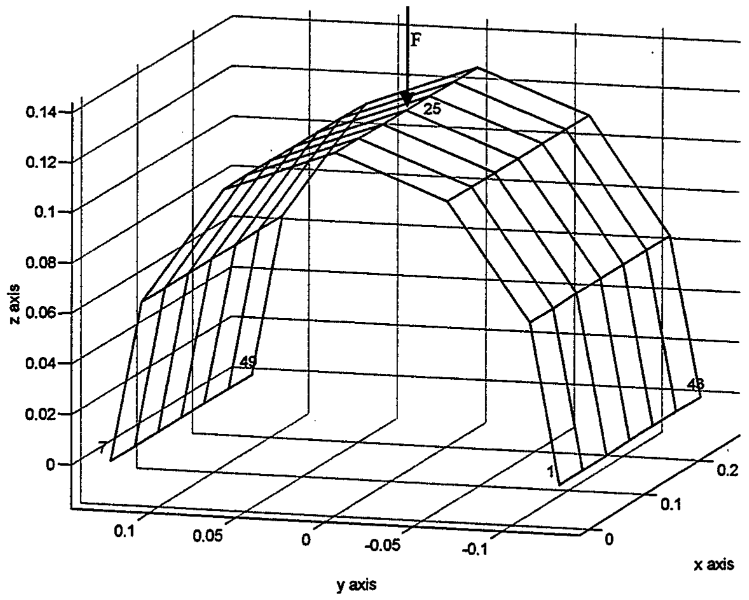


Figure 13. Mesh Structure for Pinched Cylinder.

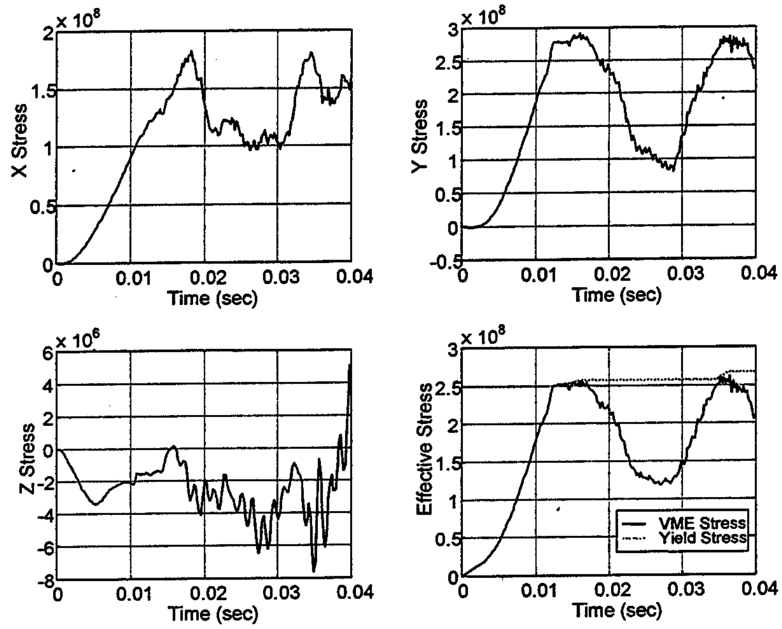


Figure 14. Stress Component Time History in Bottom Fiber: Pinched Cylinder, No Void Effects or Drilling Moment.

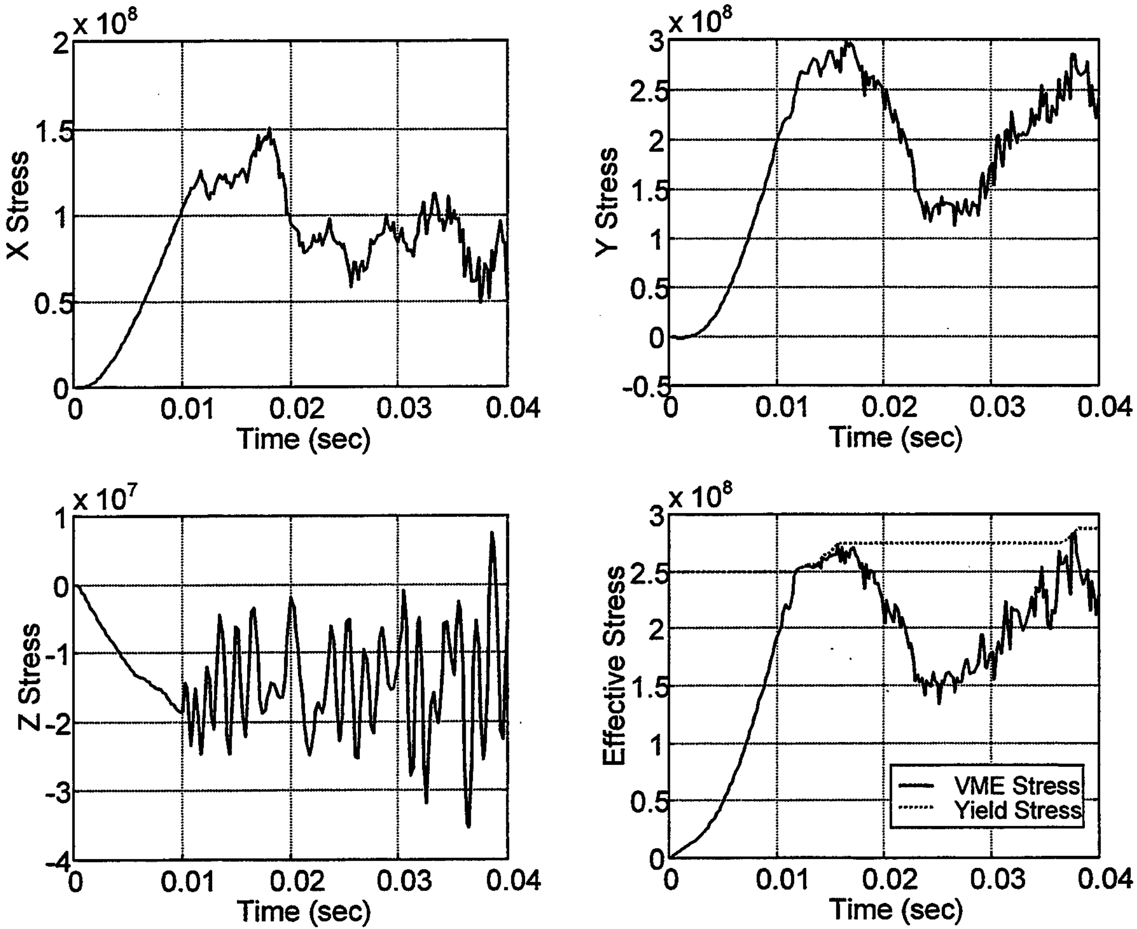


Figure 15. Stress Component Time History in Bottom Fiber: Pinched Cylinder, Void Effects and Drilling Moment Applied.

G. ELASTIC SPHERICAL CAP WITH A CENTER HOLE

The results of analysis with the shell element developed here are compared to results of the problem proposed in MacNeal and Harder [26]. The structure is a hemisphere of radius 10 units with a thickness of 0.04 units, and has an 18° hole cut in the center. Taking advantage of the symmetry in the structure, only ¼ of the structure is modeled. An eight-by-eight mesh is used, for a total of 64 elements, with four integration points through the thickness of each element. The mesh used is shown in Fig. 16. The material properties of the structure are: $E = 6.825\text{x}10^7$, $\rho = 0.001$, and $\nu = 0.3$. No void or drilling moment effects are employed in this example. Opposing forces of magnitude 2.0 are applied at each quadrant: the force at node 73 is of magnitude –1.0 and parallel to the y-axis, and the force at node one is of magnitude 1.0 and parallel to the x-axis. To obtain a representative static response using a dynamic model, the applied force begins with 0.0 magnitude at time 0, and increases linearly with a rise time of 0.16 seconds to the specified value. A calculation time step of $1\text{x}10^{-6}$ seconds is used, with termination at 0.22 seconds. The maximum deflection of node one was 0.0929, where the theoretical value is 0.0940, for a normalized displacement of 0.988. This is within the range of the results listed in MacNeal and Harder from the QUAD2 and QUAD4 elements, which are considered to be accurate for this problem [26]. These results are summarized in Table 9.

Table 9. Comparison of Results for Spherical Cap with Elastic Loading, and Results from MacNeal and Harder [26].

	New Shell Element	QUAD2	QUAD4
Normalized Displacement	0.988	0.986	1.005

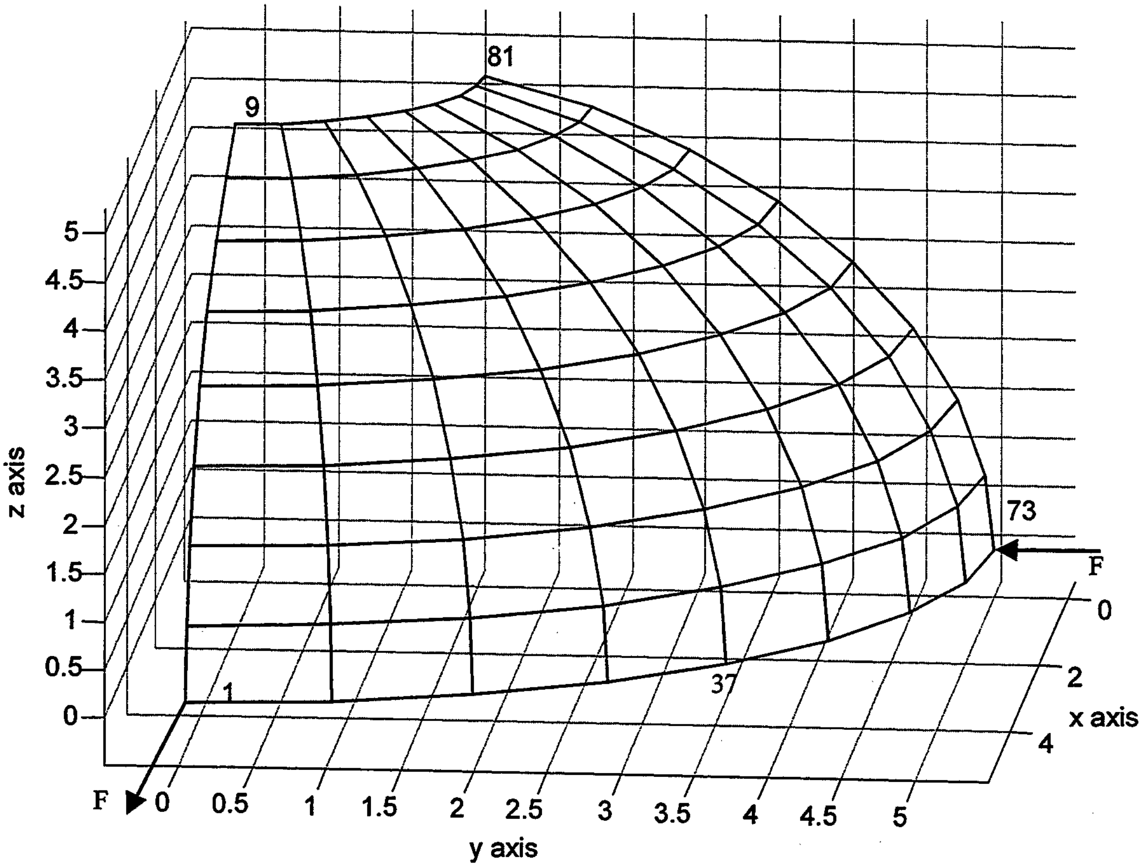


Figure 16. Mesh Structure for Spherical Cap.

H. SPHERICAL CAP WITH A CENTER HOLE, IMPACT LOADING

Key and Hoff [11] used the previous problem to test their element with an impact (step) load. The structure and matrial properties are the same, and the mesh is the same as shown in Fig. 16. The load is applied at its maximum at $t=0.0^+$ seconds, rather than ramped up. Figure 17, which plots the displacement of node one obtained from both the shell element presented here and the element developed by Key and Hoff [11], shows that the results of this element are comparable to those obtained by other elements under impact loading.

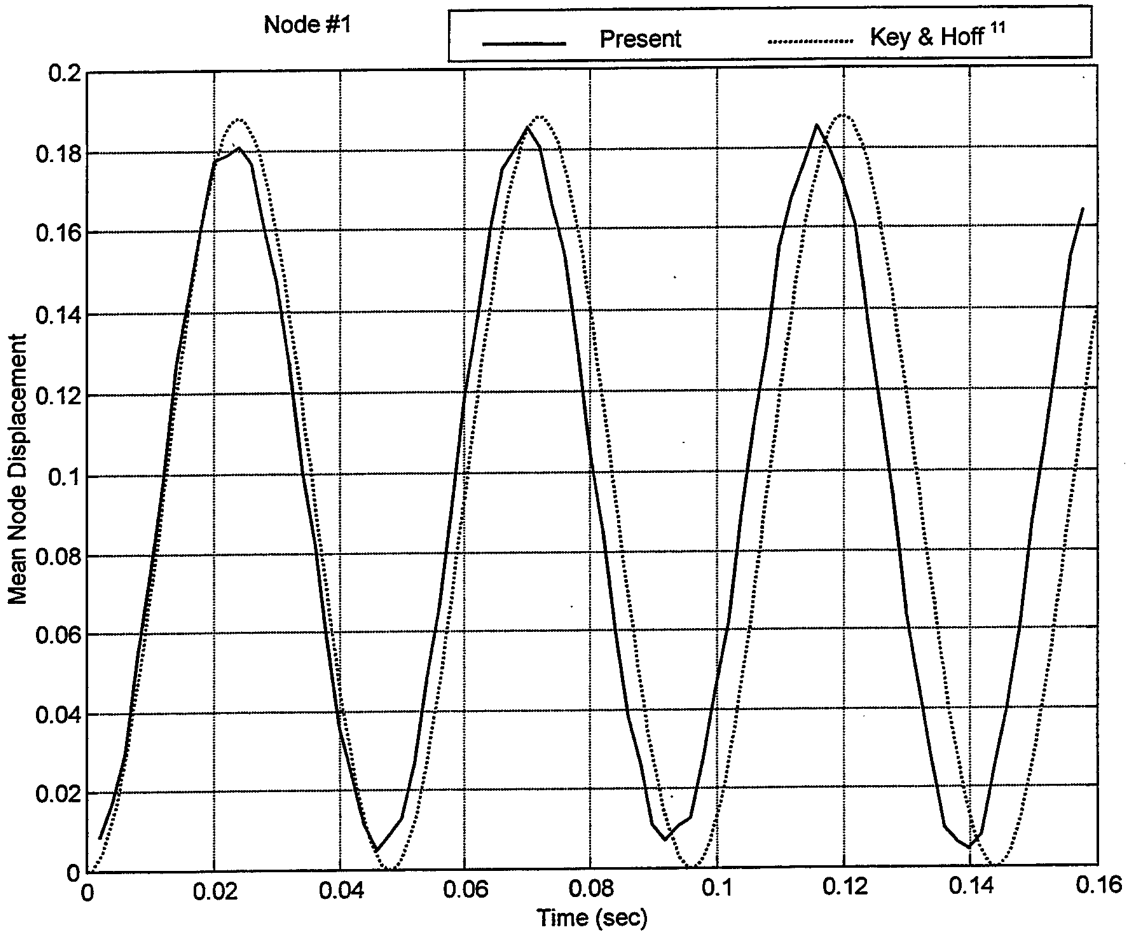


Figure 17. Node One Displacement Time History: Pinched Spherical Cap with Impact Loading.

I. SPHERICAL CAP WITH A CENTER HOLE, ELASTO-PLASTIC LOADING

The spherical cap structure shown in Fig. 16 is now used to verify the stability and convergence of the damage-constitutive equations under double curvature, and to illustrate both the effects of void growth and nucleation and a drilling moment on a more complex structure. The material properties used are the same as shown in Tables 3 and 4. The structure has a radius of 5.0 meters and a thickness of 25 cm, for a radius to thickness ratio of 20:1, commonly considered the limit of thin-shell theory. Although a force is applied at each quadrant, all four loads are directed inwards. Therefore, only one load is applied at node 37 of the mesh shown in Fig. 16. The applied load is 5.9397x10^{6} N, with a drilling moment of –7.4246x10^{5} Nm applied for the cases indicated. A calculation time step of 1x10^{-5} seconds is used, with output every 5x10^{-4} seconds and stopping at 0.2 seconds. The load is initial zero, and increases linearly to its maximum at 0.01 seconds. The results shown in Table 10 are for the element nearest the applied load, where stress is maximum. The porosity versus effective plastic strain relationship is shown in Fig. 18. Due to the small amount of plastic strain, porosity is restricted to the linear zone, as shown. Figures 19a and 19b illustrate the effective stress versus effective strain in the same element on the compressive side and tensile side, respectively. Note that as the structure returns from a peak displacement, the stress follows the elastic modulus, not the tangent modulus. This reflects the correct behavior of material: the elastic modulus is not affected by plastic deformation, only the yield stress is affected. The node displacement where the force is applied is shown in Fig. 20. This figure shows that there are at least two vibration modes in operation.

Table 10. Summary of Results for Elasto-Plastic Spherical Cap.

Peak Values for Element #25	No Voids No DM	No Voids Drilling Moment	Voids No DM	Voids and Drilling Moment
σ_{VM} (MPa)	250.26	250.76	250.26	250.75
$\varepsilon_{effective}$ (x 10^3)	1.0730	1.1021	1.0731	1.1021
σ_{xx} (MPa)	202.09	199.44	202.08	199.43
σ_{yy} (MPa)	-47.093	-58.183	-47.087	-58.192
σ_{zz} (MPa)	-5.2570	-12.433	-5.2571	-12.433
$\varepsilon^{plastic}$ (x 10^5)	2.1002	4.1795	2.1020	4.1849
Φ (Porosity) (x 10^6)	NA	NA	6.7149	12.763
Deflection (m) (Point of Loading)	0.0233	0.0246	0.0233	0.0246

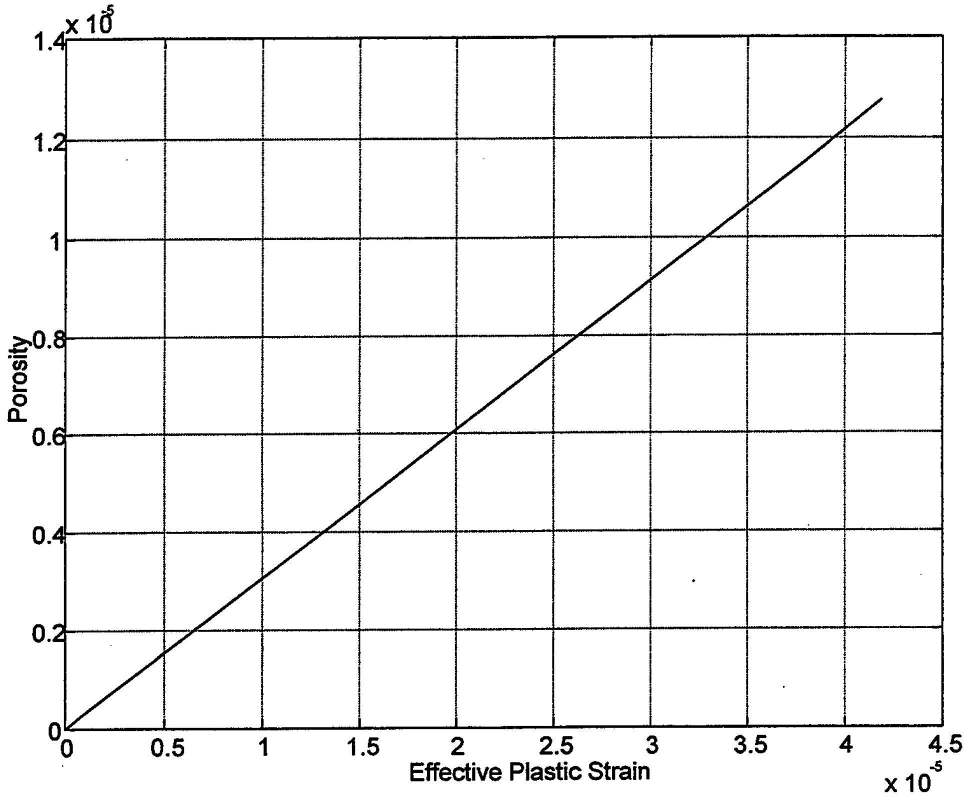


Figure 18. Porosity versus Effective Plastic Strain in Inner Fiber: Spherical Cap with Void and Drilling Moment Effects.

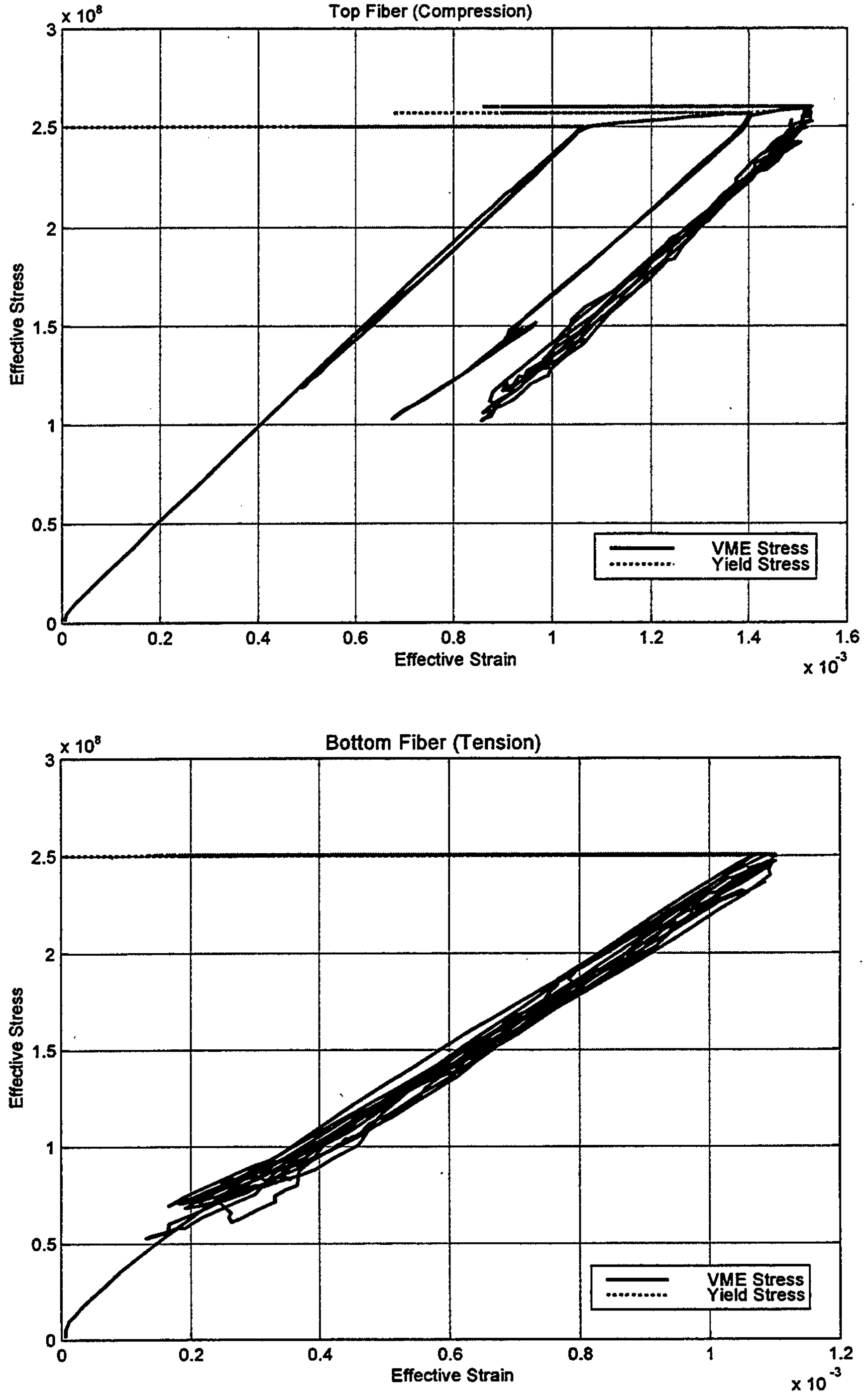


Figure 19. Comparison of Void and Drilling Moment Effects in a) Compression (Top) and b) Tension (Bottom) for Spherical Cap.

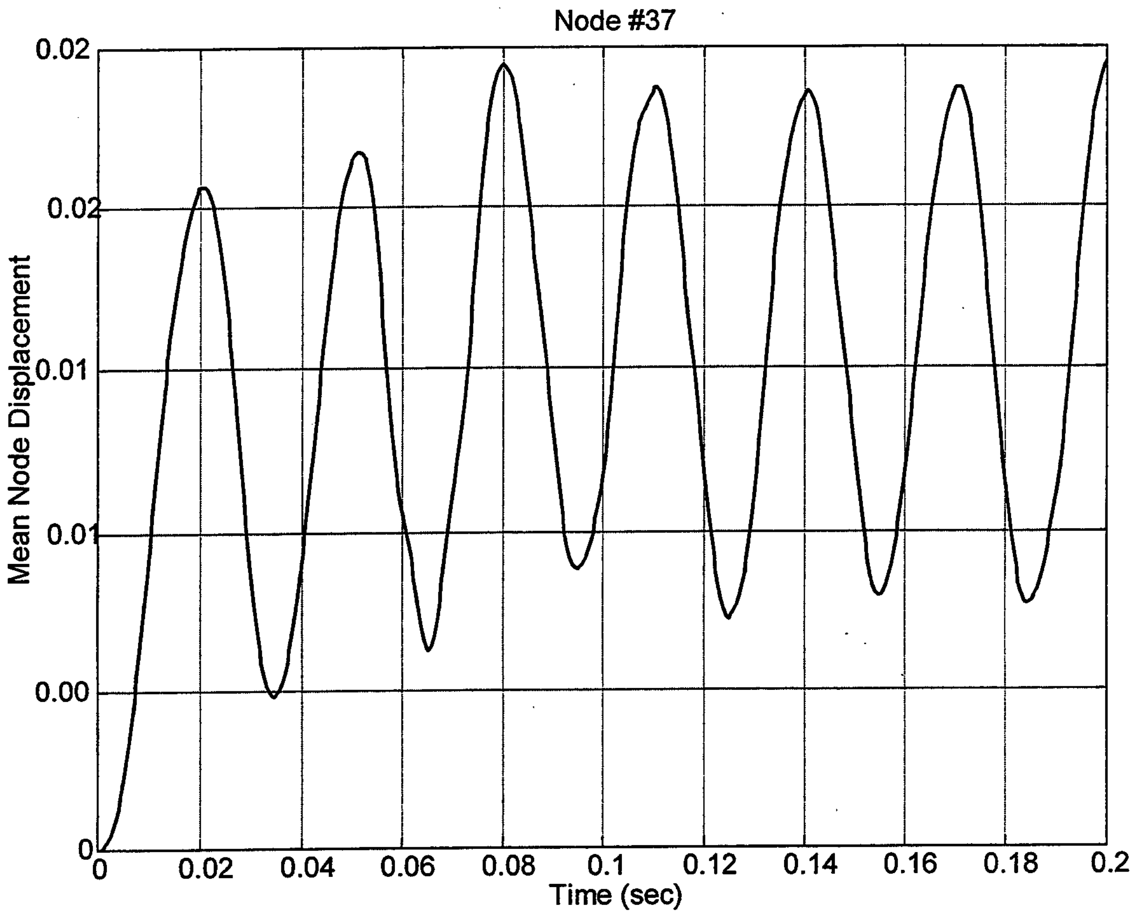


Figure 20. Contact Node Mean Displacement Time History: Spherical Cap with Void and Drilling Moment Effects.

During testing, it became apparent that this problem is not suitable for testing elasto-plastic analysis. Since the structure is concave, the entire structure quickly collapses once yielding begins. In addition, the single point used to prevent rigid-body motion also provided a stress concentration and additional yielding (the "corner" began folding over). This necessitated using a force that just causes plastic flow, but does not collapse the structure or cause yielding near the anchored node. This is illustrated by an analysis of a 5 percent greater load than used for the analysis shown in Fig. 20. The resulting node 37 displacement is shown in Fig. 21, and the deformed structure in Fig. 22. However, it is still apparent that the qualitative results obtained in the previous elasto-plastic examples carried through to this problem; both voids and the drilling moment decreased stress in fibers under tension, and increased stress in fibers under compression. Due to the small amount of plastic strain, the increase in porosity was extremely small, which minimized the effects of the voids.

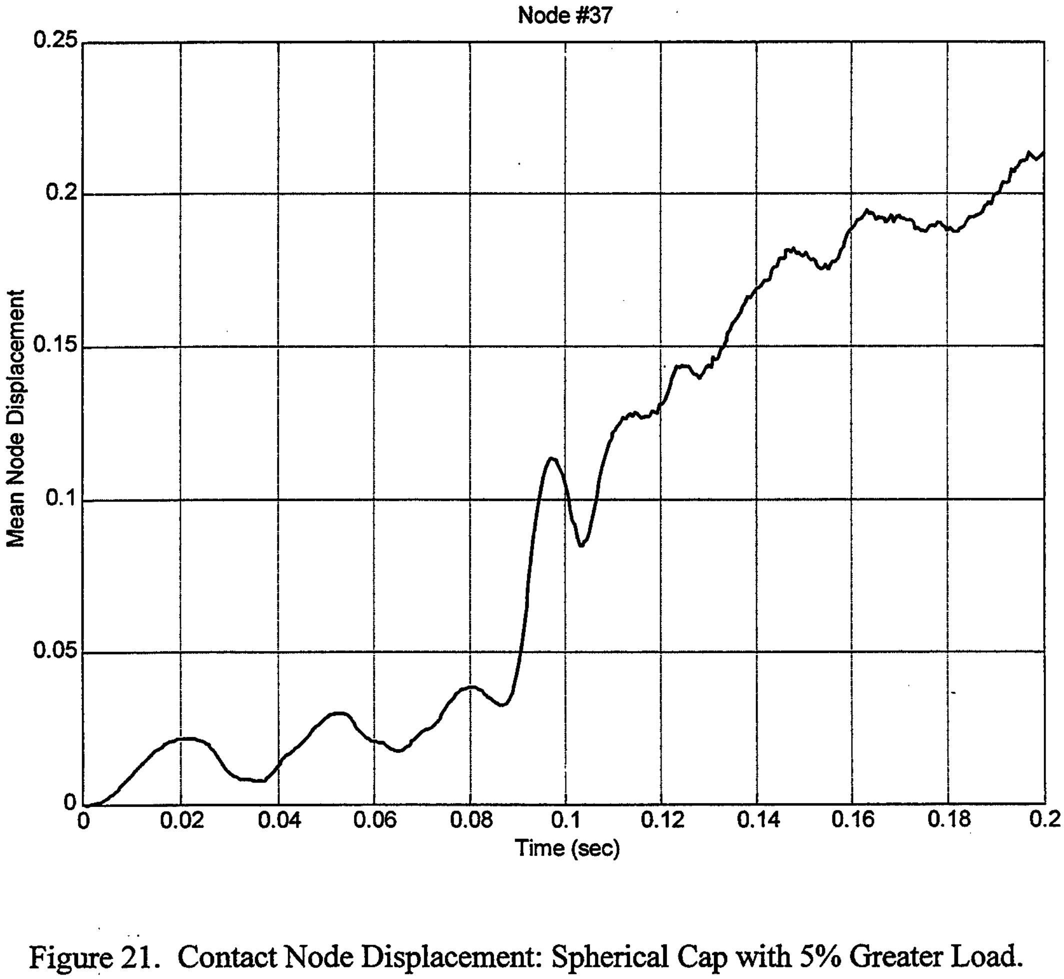


Figure 21. Contact Node Displacement: Spherical Cap with 5% Greater Load.

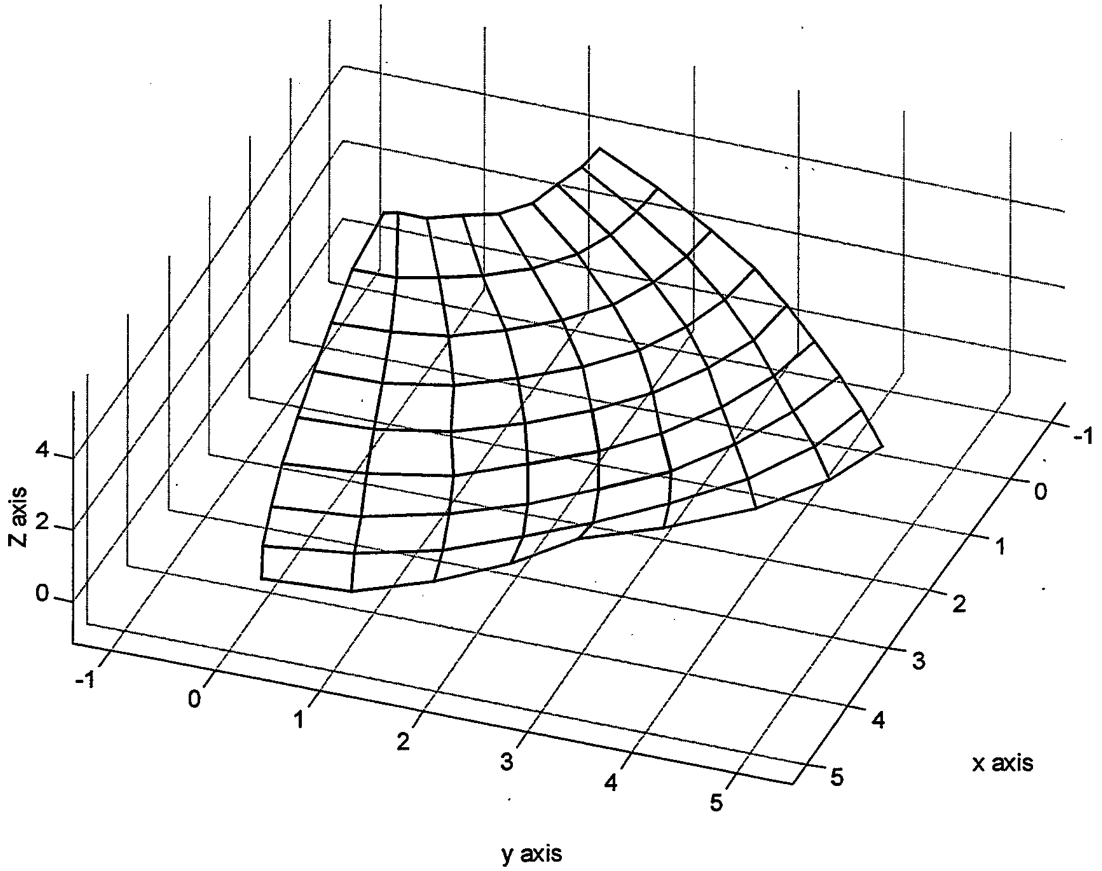


Figure 22. Deformed Structure at End of Analysis: Spherical Cap with 5% Greater Load.

VII. CONCLUSIONS AND RECOMMENDATIONS

A new shell formulation was developed for transient dynamic analysis which includes both void effects with plastic deformation and the transverse normal stress with drilling moment. The element is compatible with most shell elements which have three translation and three rotation degrees of freedom per node.

A numerically stable scheme was developed for Gurson's nonlinear constitutive equation model. The model includes both void nucleation and void growth. Furthermore, hourglass control was implemented into the algorithm to avoid spurious modes caused by the under-integration scheme.

The drilling moment was important for thick plates and shells. It induced large transverse normal stress and affected the plastic deformation. Similarly, the effect of voids on plastic deformation was not negligible. Numerical studies show that the effective plastic strain increased up to fifteen percent due to the combined effects of voids and the transverse normal stress.

The shell element presented here was tested on many cases, some of which had reference solutions for comparison. The new element gave accurate results to those problems.

Although the element formulation presented here peformed well in numerical studies, the problems studied are not in the area where this element is most likely to be useful. The number of published problems available for comparison is limited, especially in the case of dynamic plastic deformation. The solutions provided by this element in the plastic region need to be verified by experimental data, and the constants of Gurson's void model need to be determined for a variety of materials and porosity conditions. In all, there are six material property constants that must be determined before an accurate solution can be obtained: q_1, q_2, q_3, f_N, e_N, and s_N. These are not readily available, but play a vital role in the constitutive equations of this formulation.

Implementing this shell element into a general purpose finite element program will allow easier comparisons with other element formulations, including solid brick elements, and will facilitate the analysis of more complex problems. This work has already begun for a version of DYNA3D, but is not yet complete.

The method of hourglass mode control used implemented a full version of the same formulation used for the internal force calculation. Efficiency may be increased by using a simpler formulation to calculate the internal hourglass forces. Along the same lines, no attempt has been made to improve the efficiency of the code presented here so that readability may be preserved.

APPENDIX A. PREPROCESSOR IN MATLAB FOR FEA.

```
% feapre.m - Preprocessor for fea program
%     Program Limitations:
%           No input of Initial Conditions
%           Boundary Conditions must be applied manually to result file
%           Load must be applied manually to result file
%           Structure will be homogenous
%           Only 1/4 cylinder supported at this time

% Get Input data
type = input('Type of Structure (1=cylinder, 2=plate, 3=shoe box,
4=cap1, 5=cap2):');

% Structure Specific Input
if type==1
      nelx = input('Number of Elements along x-axis:');
      nelp = input('Number of Elements perpendicular to x-axis:');
      x1 = input('Lower X-axis value:');
      x2 = input('Upper X-axis value:');
   R = input('Radius:');
   theta1 = input('Starting Angle (degrees):');
   theta2 = input('Ending Angle (degress):');

elseif type==2
      nelx = input('Number of Elements along x-axis:');
      nelp = input('Number of Elements along y-axis:');
      x1 = input('Lower X-axis value:');
      x2 = input('Upper X-axis value:');
      y1 = input('Lower Y-axis value:');
      y2 = input('Upper Y-axis value:');

elseif type==3
      nelx = input('Number of Elements along x-axis:');
      nelp = input('Number of Elements Perpendicular to x-axis:');
      x1 = input('Lower X-axis value:');
      x2 = input('Upper X-axis value:');
      y1 = input('Lower Y-axis value:');
      y2 = input('Upper Y-axis value:');
      z1 = input('Lower Z-axis value:');
   z2 = input('Upper Z-axis value:');

elseif type==4
   R = input('Radius:');
   alpha = input('Phi (angle from Z-axis of inscribed circle):');
   ncap = input('Size (1=25 elem  2=42 elem  3=63 elem):');
   if ncap==1
      narc=6;
      ntheta=11;
   elseif ncap==2
      narc=8;
      ntheta=13;
   elseif ncap==3
      narc=10;
      ntheta=15;
   else
      errordlg('Bad Type');
   end
```

```
elseif type==5
   R = input('Radius:');
   alpha1 = input('Alpha1 (elevation angle to bottom ring (in
degrees):');
   alpha2 = input('Alpha2 (elevation angle to top ring (in degrees)):');
   theta1 = input('Theta1 (start angle from x-axis (in degrees)):');
   theta2 = input('Theta2 (end angle from x-axis (in degrees)):');
   nelp = input('Number of Elements along elevation angle:');
   nelx = input('Number of Elements along theta angle:');
else
      errordlg('Bad Type');
end

% Input Material and Time Data
E = input('Elastic Modulus:');
rho = input('Density (mass / unit volume):');
pois = input('Poissons Ratio:');
t = input('Thickness:');
Syp = input('Yield Stress:');
ipt = input('New(2) or Old(0) shell element (enter 2 or 0):');
stime = input('Start Time:');
etime = input('Stop Time:');
delt = input('Calculation Step Time (delta):');
ptime = input('Print Step Time:');

% If old shell element, adjust density to mass / unit area
if ipt==0
   rho = rho * t;
   ipoint = 0;
else
   ipoint = input('Number of Integration Points in Z direction:');
end

% *** Cylinder ***
if type==1
   % Calculate Node x,y,z coordinates
   theta1 = theta1 * pi / 180;
   theta2 = theta2 * pi / 180;
      nndsx = nelx+1;
      nndsr = nelp+1;
      xstep = (x2 - x1) / nelx;
      rstep = (theta2 - theta1) / nelp;
      i = 1;
      ne = nelx * nelp;
      nnodes = nndsx * nndsr;

      % Initialize Vectors
      x = zeros(nnodes,1);
      y = x;
      z = x;
      nodes = zeros(ne,4);

      for xt=x1:xstep:x2
            for r=theta1:rstep:theta2
                  x(i) = xt;
                  y(i) = -R * cos(r);
                  z(i) = R * sin(r);
                  i = i + 1;
      end
   end
   xd = x / R;
   yd = y / R;
   zd = z / R;
end

% *** Plate ***
if type==2
      nndsx = nelx+1;
      nndsy = nelp+1;
      xstep = (x2 - x1)/nelx;
      ystep = (y2 - y1)/nelp;
      i = 1;
      ne = nelx * nelp;
      nnodes = nndsx * nndsy;
      x = zeros(nnodes,1);
      y = x;
      z = x;
   nodes = zeros(ne,4);
   xd = zeros(nnodes,1);
   yd = xd;
   zd = -1 * ones(nnodes,1);

      for xt = x1:xstep:x2
            for yt=y1:ystep:y2
                  x(i) = xt;
                  y(i) = yt;
                  i = i + 1;
            end
      end
end

% *** Shoe Box ***
if type==3
   nndsx = nelx + 1;
      nndsp = nelp + 1;
      xstep = (x2 - x1)/nelx;
      ystep = (y2 - y1)/(nelp / 2);
      zstep = (z2 - z1)/(nelp / 4);
      i = 1;
      ne = nelx * nelp ;
      nnodes = nndsx * nndsp;
      x = zeros(nnodes,1);
      y = x;
   z = x;
   xd = x;
   yd = x;
   zd = x;
      nodes = zeros(ne,4);

      for xt = x1:xstep:x2
            % Front
            for zt = z1:zstep:z2
                  x(i) = xt;
         y(i) = y1;
         yd(i) = 1;
                  z(i) = zt;
                  i = i + 1;
            end
            % Top
            for yt = y1+ystep:ystep:y2
                  x(i) = xt;
                  y(i) = yt;
         z(i) = z2;
         zd(i) = -1;
                  i = i + 1;
            end
            % Back
            for zt = z2-zstep:-zstep:z1
                  x(i) = xt;
         y(i) = y2;
         yd(i) = -1;
                  z(i) = zt;
                  i = i + 1;
            end
      end
end

% Spherical Cap #1
if type==4
   % Generates FEA mesh of spherical cap using symmetry
      % Also handles nodal connectivity

      % Input Values
   theta = linspace(0,pi/2,ntheta);
   alpha = pi * alpha / 180;
      phi = linspace(0,alpha,narc);

      % Spherical Coordinates for bottom nodes
      for i=1:ntheta
      P(i,:) = [R theta(i) phi(narc)];
      bct(i) = 7;
      bcr(i) = 7;
      end

      % Spherical Coordinates for next row of nodes
      nnodes = ntheta;
      for i = 1:(ntheta-1)/2
            P(i+ntheta,:) = [R theta(2 * i) (phi(narc)-(phi(narc)-...
                              phi(narc-1))/2)];
      end

      % Spherical Coordinates for the rest of the nodes
      nnodes = nnodes + (ntheta-1)/2;
      nrow = ntheta - (ntheta-1)/2;
      for j=narc-1:-1:1
      Th = 0;
      dTh = pi / (2 * (nrow - 1));
      bct(nnodes+1)=2;
      bcr(nnodes+1)=6;
      for i=1:nrow
            P(i+nnodes,:) = [R Th phi(j)];
            Th = Th + dTh;
         end
      nnodes = nnodes + nrow;
      bct(nnodes)=1;
      bcr(nnodes)=5;
         nrow = nrow - 1;
      end
   % Change BC of top node
   bct(nnodes)=4;
   bcr(nnodes)=7;

      % Convert to Cartesian Coordinates
      for i=1:nnodes
         x(i) = P(i,1) * cos(P(i,2)) * sin(P(i,3));
         y(i) = P(i,1) * sin(P(i,2)) * sin(P(i,3));
         z(i) = P(i,1) * cos(P(i,3));
      end

      % Element Connections
      % Bottom row first
      mid = ntheta;
      top = mid + ntheta - (ntheta - 1)/2 -1;
      ne = 1;
      for i=1:(ntheta-1)/2,
         c1 = (i-1)*2 +1;
         nodes(ne,:) = [c1 (c1+1) (mid+i) (top+i)];
         nodes(ne+1,:) = [(c1+1) (c1+2) (top+1+i) (mid+i)];
         ne = ne + 2;
      end

      % Rest of elements
      nrow = ntheta - (ntheta-1)/2 - 1;
      bot = mid;
      mid = top;
      top = top + nrow + 1;
      for j=1:nrow
         for i=1:nrow
            c1 = mid + i;
            nodes(ne,:) = [c1 (bot+i) (c1+1) (top+i)];
            ne=ne+1;
      end
         bot=mid+1;
         mid=top;
         top=top+nrow;
         nrow=nrow-1;
      end
      ne = ne - 1;
end


% Spherical Cap #2 (Has Hole it top)
if type==5
      nndsx = nelx+1;
   nndsy = nelp+1;
   theta2 = theta2 * pi / 180;
   theta1 = theta1 * pi / 180;
   alpha2 = alpha2 * pi / 180;
   alpha1 = alpha1 * pi / 180;
      tstep = (theta2 - theta1)/nelx;
      astep = (alpha2 - alpha1)/nelp;
      i = 1;
      ne = nelx * nelp;
      nnodes = nndsx * nndsy;
      x = zeros(nnodes,1);
      y = x;
      z = x;
   nodes = zeros(ne,4);

      for thetat = theta1:tstep:theta2
            for alphat=alpha1:astep:alpha2
                  x(i) = R * cos(alphat) * cos(thetat);
         y(i) = R * cos(alphat) * sin(thetat);
         z(i) = R * sin(alphat);
                  i = i +1;
            end
      end
end

% Calculate Element Connectivity (will work for all shapes except cap)
if type~=4
   for i=0:(nelx - 1)
            for j=1:nelp
                  ie = (i * nelp) + j;
                  nodes(ie,1) = (i * (nelp + 1)) + j;
                  nodes(ie,2) = nodes(ie,1) + nelp + 1;
                  nodes(ie,3) = nodes(ie,2) + 1;
                  nodes(ie,4) = nodes(ie,1) + 1;
            end
      end
end

% Void content
Phi0 = input('Initial Void Content:');
q1 = input('Value for q1:');
q2 = input('Value for q2 (Enter 0 for Default):');
q3 = input('Value for q3 (Enter 0 for Default):');
fn = input('Nucleating Particle Content:');
en = input('Mean Nucleation Strain (Enter 0 for Default):');
sn = input('Nucleation Standard Deviation (Enter 0 for Default):');

% Create Stress-Strain curve
correct = 'N';
while ((correct == 'n') | (correct == 'N'))
   iseg = input('Number of Plastic Segments (0-9):');
      if (iseg ~= 0)
         for i=1:iseg
            slope(i)=input('Slope of Section:');
            strain(i)=input('Right Strain limit of Section:');
         end
      end

   strn(1) = 0;
   strss(1) = 0;
   strn(2) = Syp/E;
   strss(2) = Syp;
   if (iseg~=0)
      for i=1:iseg
         strn(2+i)=strain(i);
         strss(2+i)=strss(2+i-1) + (strn(2+i)-strn(2+i-1))*slope(i);
      end
   end

      figure(1);
      plot(strn,strss),grid,xlabel('Strain'),ylabel('Stress'),title('Mat
erial Yield Property')
   correct = input('Is the Stress-Strain Plot Correct? (y or n):','s');
```

```
end

% Fill eprop
eprop=zeros(1,40);
eprop(1) = rho;
eprop(2) = E;
eprop(3) = pois;
eprop(4) = t;
eprop(5) = Syp;
eprop(6) = q1;
eprop(7) = q2;
eprop(8) = q3;
eprop(9) = fn;
eprop(10) = en;
eprop(11) = sn;
eprop(12) = Phi0;
eprop(13) = iseg;
if (iseg > 0)
   for i=1:iseg
      eprop(12+2*i)=slope(i);
      eprop(12+2*i+1)=strain(i);
   end
end

% Display resulting structure
figure(2)
AdjView = [-37.5 30];
shownode;
rotate3d on;

% Apply Boundary Conditions
if type~=4
   bct = zeros(1,nnodes);
   bcr = bct;
end

disp('Boundary Conditions:');
disp('0 = No Constraint');
disp('1 = Constrained in X');
disp('2 = Constrained in Y');
disp('3 = Constrained in Z');
disp('4 = Constrained in X and Y');
disp('5 = Constrained in Y and Z');
disp('6 = Constrained in X and Z');
disp('7 = Constrained in X,Y, and Z');
disp('....');
if type==4,
   disp('(It is inadvisable to change BC''s for Spherical Cap)');
end

nod = input('Node to change Boundary Conditions (0 to end, -1 to
list):');
while ((nod ~= 0) & (nod <= nnodes))
   if nod < 0
      disp('Node   Trans   Rot');
      for i=1:nnodes
         disp(sprintf('  %d      %d    %d',i,bct(i),bcr(i)));
      end
   else
      disp(sprintf('Current: Translation = %d, Rotation = ',...
                  '%d',bct(nod),bcr(nod)));
      bct(nod) = input('New Translation Boundary Condition(0-7):');
      bcr(nod) = input('New Rotational Boundary Condition(0-7):');
      disp(' ');
   end
   nod = input('Node to change Boundary Condition (0 to end):');
end

% Create Load History
disp('(Pressure Load only works for Flat Plate)');
nload = input('Number of nodes to apply load to (0 = pressure load):');
ntime = input('Number of time intervals for Load Description:');
disp('Force Direction:');
disp('1 = Force along X-axis');
disp('2 = Force along Y-axis');
disp('3 = Force along Z-axis');
disp('4 = Moment about X-axis');
disp('5 = Moment about Y-axis');
disp('6 = Moment about Z-axis');

if nload == 0
      nload = nnodes * 6;
      disp('Input Pressure Time History');
      for i=1:ntime
         ldtime(i) = input('Time:');
         Pressure(i) = input('Magnitude(use neg. values for bottom',...
                        'pressure):');
      end
      F = zeros(nnodes,1);
      for elem=1:ne
         % Get Coordinates of Nodes for this element
         x1 = x(nodes(elem,1));
         x2 = x(nodes(elem,2));
         x3 = x(nodes(elem,3));
         x4 = x(nodes(elem,4));
         y1 = y(nodes(elem,1));
         y2 = y(nodes(elem,2));
         y3 = y(nodes(elem,3));
         y4 = y(nodes(elem,4));
         z1 = z(nodes(elem,1));
         z2 = z(nodes(elem,2));
         z3 = z(nodes(elem,3));
         z4 = z(nodes(elem,4));

         % Calculate length of sides, then area
         L1 = sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2);
         L2 = sqrt((x3-x2)^2+(y3-y2)^2+(z3-z2)^2);
         L3 = sqrt((x4-x3)^2+(y4-y3)^2+(z4-z3)^2);
         L4 = sqrt((x1-x4)^2+(y1-y4)^2+(z1-z4)^2);
         dot1 = (x1-x2)*(x3-x2)+(y1-y2)*(y3-y2)+(z1-z2)*(z3-z2);
         dot2 = (x1-x4)*(x3-x4)+(y1-y4)*(y3-y4)+(z1-z4)*(z3-z4);
         Area = 0.5 * (sqrt(L1^2 * L2^2 - dot1^2) + sqrt(L3^2 * ...
                  L4^2 - dot2^2));
         % Calculate Force per unit pressure (assumes square elements)
         F(nodes(elem,1)) = F(nodes(elem,1)) + Area / 4;
         F(nodes(elem,2)) = F(nodes(elem,2)) + Area / 4;
         F(nodes(elem,3)) = F(nodes(elem,3)) + Area / 4;
         F(nodes(elem,4)) = F(nodes(elem,4)) + Area / 4;
      end
      % Create Node,Force,Time Vectors
      nc = 1;
      for  n=1:nnodes,
         loadnode(nc) = n;
         loadnode(nc+1) = n;
         loadnode(nc+2) = n;
         loadnode(nc+3) = n;
         loadnode(nc+4) = n;
         loadnode(nc+5) = n;
         loaddir(nc) = 1;
         loaddir(nc+1) = 2;
         loaddir(nc+2) = 3;
         loaddir(nc+3) = 4;
         loaddir(nc+4) = 5;
         loaddir(nc+5) = 6;
         loadtime(nc,:) = ldtime;
         loadtime(nc+1,:) = ldtime;
         loadtime(nc+2,:) = ldtime;
         loadtime(nc+3,:) = ldtime;
         loadtime(nc+4,:) = ldtime;
         loadtime(nc+5,:) = ldtime;
         loadmag(nc,:) = xd(n) * F(n) * Pressure;
         loadmag(nc+1,:) = yd(n) * F(n) * Pressure;
         loadmag(nc+2,:) = zd(n) * F(n) * Pressure;
         loadmag(nc+3,:) = loadmag(nc,:) * t / 2;
         loadmag(nc+4,:) = loadmag(nc+1,:) * t / 2;
         loadmag(nc+5,:) = loadmag(nc+2,:) * t / 2;

         nc = nc + 6;
      end

else
   disp('Input Load/Time/Magnitude History');
   for n=1:nload
      loadnode(n) = input('Node Number:');
      loaddir(n) = input('Load Direction:');
      for nt = 1:ntime
         loadtime(n,nt) = input('Time:');
         loadmag(n,nt) = input('Magnitude:');
      end
   end
end

% Create draft input file
F = fopen('input.in','w');

fprintf(F,'%d 1 %d %d 0 %d 0 \r\n',nnodes,nload,ntime,ipt);
fprintf(F,'%e %e %e %e \r\n',stime,etime,delt,ptime);
fprintf(F,'0 0 0 0 0 0 \r\n');
fprintf(F,'0 %d 0 0 0 \r\n',ne);
fprintf(F,'0 0 0 0 \r\n');
fprintf(F,'0 0\r\n0 0\r\n%d\r\n1 1 \r\n',ipoint);
for i=1:4
   fprintf(F,'%e %e %e %e %e %e %e %e %e %e\r\n',...
                  eprop((1+(i-1)*10):(i*10)));
end
for i=1:nnodes
      fprintf(F,'%d %d %d %e %e %e \r\n',...
                  i,bct(i),bcr(i),x(i),y(i),z(i));
end
```

```
for i=1:ne
      fprintf(F,'%d 1 %d %d %d %d 0 \r\n',i,nodes(i,:));
end
for i=1:nload
   fprintf(F,'%d %d\r\n',loadnode(i),loaddir(i));
   for nt = 1:ntime
      fprintf(F,'%e %e\r\n',loadtime(i,nt),loadmag(i,nt));
   end
end

fclose(F);
```

APPENDIX B. POST-PROCESSOR IN MATLAB FOR FEA.

```
% fea_disp.m
% P. McDermott
%
% Uses GUI to read and display output from fea program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Scrn = get(0,'ScreenSize');
ht = round(0.9 * Scrn(4) / 2 - 50);
wd = round(0.9 * (Scrn(3) - 120) / 2);

hf = figure('NumberTitle','off','Name','FEA','MenuBar','none',...
            'Position',[20 (Scrn(4) - 500) 120 460],'Color','b');

hfs = figure('NumberTitle','off','Name','Structure
View','Visib','off',...
            'Position',[160 (Scrn(4)-70-ht) wd ht]);

hfd = figure('NumberTitle','off','Name','Displacement','Visib','off',...
            'Position',[(wd+180) (Scrn(4) - 70-ht) wd ht]);

hlast = 0;
AdjView = [-37.5 30];
flagd = 1;
flagv = 0;
flaga = 0;
flags = 0;

File = 'output.fea';
File2 = 'output.fes';

P = pwd;
Path = strcat(P,'\');

hc_change = uicontrol(hf,'Style','push','String','Change File',...
            'Position',[20 380 80 20],...
            'CallBack',[...
            '[File1 Path1] = uigetfile(''*.fea'',''Load FEA Output
File'');',...
            'if File1 ~= 0,'...
                  'File = File1;'...
         'Path = Path1;'...
         'File2 = File;'...
         'File2(length(File2)) = ''s'';'...
                  'set(hc_file,''String'',File);'...
            'end']);

hc_load = uicontrol(hf,'Style','check','String','Load File',...
            'Position',[20 420 80 20],...
            'CallBack',[...
         'FilePath = [Path File];',...
         'FilePath2 = [Path File2];',...
                  'parse;',...
                  'set(hc_clear,''Enable'',''on'');',...
                  'set(hc_disp,''Enable'',''on'');',...
                  'set(hc_velo,''Enable'',''on'');',...
                  'set(hc_acc,''Enable'',''on'');',...
                  'set(hc_show,''Enable'',''on'');',...
                  'set(hc_load,''Value'',1,''Enable'',''off'');']);

hc_clear = uicontrol(hf,'Style','push','String','Clear
Data','Enable','off',...
            'Position',[20 60 80 20],'CallBack',[...
            'clear n* x* y* z* i* s* t* v* a* fp factor d etime',…
                  'dt m time;',...
            'set(hfd,''Visib'',''off'');',...
            'if (flagv>0),set(hfv,''Visib'',''off'');end;',...
            'if (flaga>0),set(hfa,''Visib'',''off'');end;',...
            'set(hfs,''Visib'',''off'');',...
            'flagd = 1; flagv = 0; flaga = 0; flags = 0;',...
            'set(hc_load,''Value'',0,''Enable'',''on'');',...
            'set(hc_disp,''Enable'',''off'');',...
            'set(hc_velo,''Enable'',''off'');',...
            'set(hc_acc,''Enable'',''off'');',...
            'set(hc_show,''Enable'',''off'');',...
            'set(hc_replay,''Enable'',''off'');',...
            'hlast = 0;',...
            'set(hc_mview,''Enable'',''off'');',...
            'set(hc_clear,''Enable'',''off'');']);

hc_close = uicontrol(hf,'Style','push','String','Close',...
                  'Callback',[...
                        'close(hfd);',...
                        'if (flaga>0),close(hfa);end;',...
                        'if (flagv>0),close(hfv);end;',...
                        'close(hfs);',...
                        'clear;',...
                        'close;'],...
                  'Position',[20 20 80 20]);

hc_disp = uicontrol(hf,'Style','push','String','Displace','Position',…
            [20 260 80 20],...
                  'Enable','off',...
                  'Callback',[...
                  'if (flags)',...
                        'AdjView = get(hax,''View'');',...
                  'end;',...
                  'figure(hfd);',...
                  'displace;',...
                  'set(hc_replay,''Enable'',''on'');',...
                  'flagd = 0;',...
                  'hlast = hfd;']);

hc_mview = uicontrol(hf,'Style','push','String', 'Adjustment',...
                  'Position',[20 100 80 20],...
                  'Enable','off',...
                  'CallBack',[...
                        'figure(hfs);',...
                        'mmview3d;']);

hc_file = uicontrol(hf,'Style','text','Position',[20 340 80 20],...
                  'String',File);
```

```
hc_velo = uicontrol(hf,'Style','push','String','Graphs','Position',[20
220 80 20],...
                  'Enable','off',...
         'Callback',[...
            'flagv=1;',...
            'control;']);

hc_acc = uicontrol(hf,'Style','push','String','Stress','Position',[20
180 80 20],...
                  'Enable','off',...
            'Callback',[...
                  'if (flags)',...
                              'AdjView = get(hax,''View'');',...
                        'end;',...
            'figure(hfd);',...
            'stresscolor;',...
            'flagd = 0;',...
                  'set(hc_replay,''Enable'',''on'');',...
                     'hlast = hfd;']);

hc_replay = uicontrol(hf,'Style','push','String','Replay',...
                  'Position',[20 140 80 20],...
                  'Enable','off',...
                  'Callback',[...
                        'figure(hlast);',...
                        'movie(m);']);

hc_show = uicontrol(hf,'Style','push','String','Show',...
                  'Position',[20 300 80 20],...
                  'Enable','off',...
                  'Callback',[...
                        'figure(hfs);',...
                        'shownode;',...
                        'set(hc_mview,''Enable'',''on'');',...
                        'flags = 1;',...
                        'hax = gca;']);

%%%%%%%%%%%%%%%%%%%%%% End FEA_DISP.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% PARSE.M
%     This m-file reads the output file generated by fea.f

%     and parses it into matlab usable variables which will be used

%     by another program to graphically display the data in various

%     formats

%                                  McDermott 1999
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Create Progress Box and 1st Message
hprog = figure('NumberTitle','off','Name','Load File
Progress','MenuBar','none',...
   'Position',[(Scrn(3)/2 - 80) (Scrn(4)/2 - 40) 260 80],'Color','y');
hmsg = uicontrol('Style','text','Position',…
      [20 45 220 20],'BackgroundColor','y',...
   'String','Opening Files...');
hmsg2 = uicontrol('Style','text','Position',…
      [20 15 220 20],'BackgroundColor','y',...
      'String',FilePath);
drawnow;

fp=fopen(FilePath,'r');
fp2 = fopen(FilePath2,'r');

top=fscanf(fp,'%e',33);

% Update Progress Box
set(hmsg,'String','Loading Displacement');
drawnow;

% split into var names

set(hmsg2,'String','Input & eprop');
drawnow;

nnodes=top(1);
nmat=top(2);
nload=top(3);
loadtime=top(4);
initc=top(5);
stime=top(8);
etime=top(9);
dt=top(11);
ne=top(19);
ipoint = top(31);

for i=1:nmat,
      eprop = fscanf(fp,'%e',40);
end;

for i=1:(nnodes),
      fscanf(fp,'%e',3);            % dummy index
      x(i)=fscanf(fp,'%e',1);       % x coordinates of nodes
      y(i)=fscanf(fp,'%e',1);       % y coordinates of nodes
      z(i)=fscanf(fp,'%e',1);       % z coordinates of nodes
end;
```

```
for i=1:ne,
      fscanf(fp,'%e',2);
      nodes(i,:)=fscanf(fp,'%e',4)';      % associate nodes w/ elements
      fscanf(fp,'%e',1);
end;

%initial conditions
fscanf(fp,'%e',((3*(6*nnodes))+(2*nload*(loadtime+1))));
nsteps=round((etime-stime)/dt);
time = [];
temp = nsteps;
for t=1:nsteps,
   set(hmsg2,'String',['Time Step: ' num2str(t) ' / ' num2str(nsteps)]);
   drawnow;
   test=fscanf(fp,'%e',1);
   if (test > 0.0),
      time(t) = test;
            for i=1:nnodes,
            fscanf(fp,'%e',1);            %dummy index
                  xu(i,t)=fscanf(fp,'%e',1);
                  xv(i,t)=fscanf(fp,'%e',1);
                  xa(i,t)=fscanf(fp,'%e',1);
                  fscanf(fp,'%e',1);
                  yu(i,t)=fscanf(fp,'%e',1);
                  yv(i,t)=fscanf(fp,'%e',1);
                  ya(i,t)=fscanf(fp,'%e',1);
                  fscanf(fp,'%e',1);
                  zu(i,t)=fscanf(fp,'%e',1);
                  zv(i,t)=fscanf(fp,'%e',1);
                  za(i,t)=fscanf(fp,'%e',1);
         fscanf(fp,'%e',1);
         thxu(i,t)=fscanf(fp,'%e',1);
         thxv(i,t)=fscanf(fp,'%e',1);
         thxa(i,t)=fscanf(fp,'%e',1);
         fscanf(fp,'%e',1);
         thyu(i,t)=fscanf(fp,'%e',1);
         thyv(i,t)=fscanf(fp,'%e',1);
         thya(i,t)=fscanf(fp,'%e',1);
         fscanf(fp,'%e',1);
         thzu(i,t)=fscanf(fp,'%e',1);
         thzv(i,t)=fscanf(fp,'%e',1);
         thza(i,t)=fscanf(fp,'%e',1);
      end;
   else
      % EOF reached, terminate loop storing maximum number of time steps
      t = nsteps;
   end;
end;

nsteps = size(zu,2);
fclose(fp);

% Update Progress box
set(hmsg,'String','Calculating Scales');
set(hmsg2,'String','  ');
drawnow;

% Autoscale vectors
dx = abs(max(x) - min(x));
dy = abs(max(y) - min(y));
```

```
dz = abs(max(z) - min(z));

% Check for flat structures
if dx == 0
      dx = 1;
end

if dy == 0
      dy = 1;
end

if dz == 0
      dz = 1;
end

scale(1) = dx / max(max(abs(xu)));
scale(2) = dy / max(max(abs(yu)));
scale(3) = dz / max(max(abs(zu)));

dscale = min(scale) * 0.25;
if dscale < 1
      dscale = 1;
end

xu = xu * dscale;
yu = yu * dscale;
zu = zu * dscale;

scale(1) = dx / max(max(abs(xv)));
scale(2) = dy / max(max(abs(yv)));
scale(3) = dz / max(max(abs(zv)));

vscale = min(scale) * 0.75;
if vscale < 1
      vscale = 1;
end

xv = xv * vscale;
yv = yv * vscale;
zv = zv * vscale;

scale(1) = dx / max(max(abs(xa)));
scale(2) = dy / max(max(abs(ya)));
scale(3) = dz / max(max(abs(za)));

ascale = min(scale) * 0.5;
if ascale < 1
      ascale = 1;
end

xa = xa * ascale;
ya = ya * ascale;
za = za * ascale;
```

```
% Allocate arrays for stress-strain data
stress = zeros(ipoint*ne*6,nsteps);
strain = stress;
plastrn = stress;
void = zeros(ipoint*ne,nsteps);
yieldstrs = void;
effplstrn = void;
vmstress = void;
effstrain = void;
estrain = void;
%pstrain = void;

% Update Progress Box
set(hmsg,'String','Loading Stress & Strain');
drawnow;

% Read Stress-Strain Data
for t=1:nsteps,
   test = fscanf(fp2,'%s',1);
   set(hmsg2,'String',['Time Step: ' num2str(t) ' / ' num2str(nsteps)]);
   drawnow;

   if (test > 0.0),
      t1=fscanf(fp2,'%e',1);
      for elem=1:ne,
         fscanf(fp2,'%s',1);
            fscanf(fp2,'%d',1);
         for point=1:ipoint
            index = (elem-1)*6*ipoint + (point-1)*6;
            fscanf(fp2,'%d',1);
            for i = 1:6,
               ind = i + index;
               stress(i + index,t)=fscanf(fp2,'%e',1);
               strain(i + index,t)=fscanf(fp2,'%e',1);
               plastrn(i + index,t)=fscanf(fp2,'%e',1);
            end
            index1 = (elem-1)*ipoint + point;
            yieldstrs(index1,t) = fscanf(fp2,'%e',1);
            void(index1,t) = fscanf(fp2,'%e',1);
            effplstrn(index1,t) = fscanf(fp2,'%e',1);
         end
      end
   else
      t = nsteps;
   end
end

% Calculate Von-Misses Stress and an equivalent total strain value

% Update Progress Box
set(hmsg,'String','Calculating Effective');
set(hmsg2,'String','Stress and Strain...');
drawnow;
fclose(fp2);

% Adjust for total strain
%totalstrn = strain + plastrn;
```

```
% Get Effective stress and strain (doesn't work with multiple materials)
for elem=1:ne,
   for point=1:ipoint,
      index = (elem-1)*6*ipoint + (point-1)*6 + 1;
      index1 = (elem-1)*ipoint + point;
      vmstress(index1,:)=sqrt(0.5*((stress(index,:) - …
            stress(index+1,:)).^2 + ...
            ( stress(index+1,:) - stress(index+2,:)).^2 +…
            ( stress(index+2,:) - stress(index,:)).^2 + 6 *…
            (stress(index+3,:).^2 + stress(index+4,:).^2 …
            + stress(index+5,:).^2)));


      estrain(index1,:)=sqrt(2*((strain(index,:)-strain(index+1,:)).^2…
            +(strain(index+1,:)-strain(index+2)).^2 + (strain(index+2)…
            - strain(index,:)).^2) + 3*(strain(index+3).^2 + …
            strain(index+4).^2 + strain(index+5).^2))/(2*(1+eprop(3)));
      end
end

close(hprog);

%%%%%%%%%%%%%%%%%%%%  End of PARSE.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Control.m
%     Display Von-Misses Stress and Yield Stress curves
%                                                    McDermott 1999
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hfv = figure('NumberTitle','off','Name','Graph Control','Visib','on',...
            'MenuBar','none','Color','b','Position',[160 50 wd ht]);

hfa = figure('NumberTitle','off','Name','Plot Window','Visib','off',...
            'Position',[(wd+180) 50 wd ht]);

elementstr = ['1|'];
for i=2:ne
   elementstr = strcat(elementstr,num2str(i));
   elementstr = strcat(elementstr,'|');
end

pointstr = ['1|'];
for i=2:ipoint
   pointstr = strcat(pointstr,num2str(i));
   pointstr = strcat(pointstr,'|');
end

nodestr = ['1|'];
for i=2:nnodes
   nodestr = strcat(nodestr,num2str(i));
   nodestr = strcat(nodestr,'|');
end

typestr = ['Stress vs Strain|Stress vs Time|Strain vs time|Porosity vs
Time|',...
      'Porosity vs Strain|Sigma zz vs Thickness|Sigma xx vs Time|',...
      'Sigma yy vs Time|Sigma zz vs Time|Sigma Multiplot|',…
      'X Disp. vs Time|Y Disp. vs Time|Z Disp. vs Time|',…
      'Theta x vs Time|Theta y vs Time|',...
      'Theta z vs Time|MeanDisp vs Time|Plastic Strain|Elastic Strain'];

% Start at Bottom
hv_plot = uicontrol(hfv,'Style','push','Position',…
      [(wd/2 - 40) 20 80 20],'String','Plot','Callback',…
      'flaga=1;drawstress;');

hv_typetxt = uicontrol(hfv,'Style','text','Position',[ 20 60 120 20],...
   'String','Graph Type:');

hv_type = uicontrol(hfv,'Style','popupmenu','Position',…
            [ 160 60 160 20 ],'String',typestr,'Value',1,…
            'Callback','type = get(hv_type,''Value'');');
type = 1;

hv_pointtxt = uicontrol(hfv,'Style','text','Position',…
            [ 20 100 120 20 ],'String','Integration Point:');

hv_point = uicontrol(hfv,'Style','popupmenu','Position',…
            [ 280 100 40 20 ],'String',pointstr,'Value',1,…
            'Callback','point = get(hv_point,''Value'');');
point = 1;

hv_elemtxt = uicontrol(hfv,'Style','text','Position',…
            [ 20 140 120 20 ],'String','Element Number:');
```

```
hv_elem = uicontrol(hfv,'Style','popupmenu','Position',…
            [280 140 40 20 ],'String',elementstr,'Value',1,…
            'Callback','selem = get(hv_elem,''Value'');');
selem = 1;

hv_nodetxt = uicontrol(hfv,'Style','text','Position',…
            [20 180 120 20 ],'String','Node Number:');

hv_node = uicontrol(hfv,'Style','popupmenu','Position',…
            [280 180 40 20],'String',nodestr,'Value',1,…
            'Callback','snode = get(hv_node,''Value'');');
snode = 1;

hv_slidtxt = uicontrol(hfv,'Style','text','Position',[20 260 120 20],...
   'String','Time Slider Value: ');

hv_slidtxt2 = uicontrol(hfv,'Style','text','Position',…
            [160 260 80 20],'String','0');

hv_slidtxt3 = uicontrol(hfv,'Style','text','Position',…
            [240 260 80 20],'String','  Seconds');

hv_slid = uicontrol(hfv,'Style','slider','Position',…
            [20 220 (wd - 40) 20],'Min',1,'Max',nsteps,'Value',1,…
            'Callback',...
      ['tstp = round(get(hv_slid,''Value''));',...
       'set(hv_slidtxt2,''String'',num2str(time(tstp)));']);
tstp = 1;

set(hfv,'Visib','on');

%%%%%%%%%%%%%%%%%%%%%%%%%% End of CONTROL.M %%%%%%%%%%%%%%%%%%%%%%%%
```

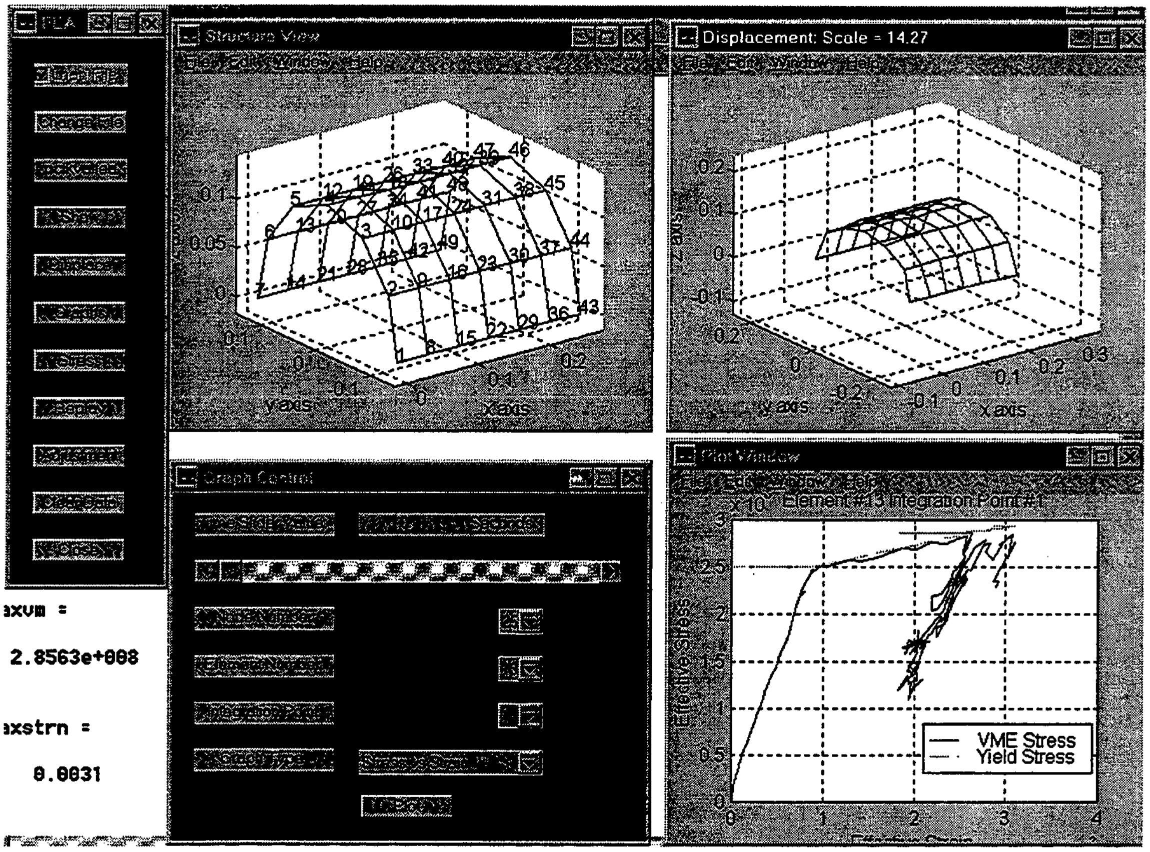
```
% displace.m
%     Display the element and its displacement at each time step
%                                                McDermott 1999
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% plot the element(s)
if flagd
      zmax=max(z)+max(max(zu))+.1;
      ymax=max(y)+max(max(yu))+.1;
      xmax=max(x)+max(max(xu))+.1;
      zmin=min(z)+min(min(zu))-.1;
      ymin=min(y)+min(min(yu))-.1;
      xmin=min(x)+min(min(xu))-.1;
      AxMax = [xmin,xmax,ymin,ymax,zmin,zmax];
end;

m=moviein(nsteps);

% Display scaling factor in title bar
dispstring = ['Displacement: Scale = ' num2str(dscale,'%1.4g')];
set(gcf,'Name',dispstring);

for t=1:nsteps
      % display global node numbers
      cla,xlabel('x axis'),ylabel('y axis'),zlabel('Z axis'),grid on,...
      set(gca,'Box','off'),hold on;

      for i=1:ne
            x1=x(nodes(i,1))+xu(nodes(i,1),t);
            x2=x(nodes(i,2))+xu(nodes(i,2),t);
            x3=x(nodes(i,3))+xu(nodes(i,3),t);
            x4=x(nodes(i,4))+xu(nodes(i,4),t);
            y1=y(nodes(i,1))+yu(nodes(i,1),t);
            y2=y(nodes(i,2))+yu(nodes(i,2),t);
            y3=y(nodes(i,3))+yu(nodes(i,3),t);
            y4=y(nodes(i,4))+yu(nodes(i,4),t);
            z1=z(nodes(i,1))+zu(nodes(i,1),t);
            z2=z(nodes(i,2))+zu(nodes(i,2),t);
            z3=z(nodes(i,3))+zu(nodes(i,3),t);
            z4=z(nodes(i,4))+zu(nodes(i,4),t);

            xvec=[x1,x2,x3,x4,x1];
            yvec=[y1,y2,y3,y4,y1];
            zvec=[z1,z2,z3,z4,z1];

            plot3(xvec,yvec,zvec),view(AdjView),axis(AxMax);
      end;
      hold off;
      m(:,t)=getframe;
end;
movie(m);
%%%%%%%%%%%%%%%%%%%%%%%  End of DISPLACE.M %%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% shownode.m
%     Display the element
%                                   McDermott 1999
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Remove previous contents
cla;

% plot the element(s)
offset = 0.1 * (max(z) + max(y) + max(x))/3;

zmax=max(z)+offset;
ymax=max(y)+offset;
xmax=max(x)+offset;
zmin=min(z)-offset;
ymin=min(y)-offset;
xmin=min(x)-offset;
AxMax = [xmin,xmax,ymin,ymax,zmin,zmax];

xlabel('x axis'),ylabel('y axis'),zlabel('z axis'),...
    axis(AxMax),view(AdjView),grid,hold on;

for i=1:ne
      x1=x(nodes(i,1));
      x2=x(nodes(i,2));
      x3=x(nodes(i,3));
      x4=x(nodes(i,4));
      y1=y(nodes(i,1));
      y2=y(nodes(i,2));
      y3=y(nodes(i,3));
      y4=y(nodes(i,4));
      z1=z(nodes(i,1));
      z2=z(nodes(i,2));
      z3=z(nodes(i,3));
      z4=z(nodes(i,4));

      xvec=[x1,x2,x3,x4,x1];
      yvec=[y1,y2,y3,y4,y1];
      zvec=[z1,z2,z3,z4,z1];

      plot3(xvec,yvec,zvec);
end;

d=offset * 0.25;
for i=1:nnodes
      text(x(i)+d, y(i)+(2*d),z(i)+d,num2str(i));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%% End of SHOWNODE.M %%%%%%%%%%%%%%%%%%%%%
```

```
% drawstress.m - plots Von-Mises stress and Yield stress
%                                         McDermott 1999
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(hfa);
set(hfa,'Visib','on');
flaga = 1;

% tstp = time vector index (not all plots need)
% point = selected integration point
% selem = selected element
% type = Graph Type selected:
%           1 = Stress vs Strain
%           2 = Stress vs Time
%           3 = Strain vs Time
%           4 = Porosity vs Time
%           5 = Porosity vs Plastic Strain
%           6 = Z Stress vs Thickness (at time(tstp))
%           7 = X Stress vs Time
%           8 = Y Stress vs Time
%           9 = Z Stress vs Time
%           10 = Stress Multiplot (all 4 stress plots vs Time)
%           11 = X Displacement vs Time
%           12 = Y Displacement vs Time
%           13 = Z Displacement vs Time
%           14 = Theta X vs Time
%           15 = Theta Y vs Time
%           16 = Theta Z vs Time
%           17 = Mean Displacement vs Time
%           18 = Plastic Strain vs Time
%           19 = Elastic Strain vs Time

index = (selem-1)*6*ipoint + (point-1)*6;
index1 = (selem-1)*ipoint + point;

switch type
      case 1,
            titlestr = ['Element #' num2str(selem),…
                  ' Integration Point #' num2str(point)];

plot(effstrain(index1,:),vmstress(index1,:),effstrain(index1,:),...
         yieldstrs(index1,:),'--'),grid,xlabel('Effective Strain'),...
         ylabel('Effective Stress'),title(titlestr),…
         legend('VME Stress','Yield Stress')
      maxvm = max(vmstress(index1,:))
      maxstrn = max(effstrain(index1,:))

   case 2,
      titlestr = ['Element #' num2str(selem) ' Integration Point #',…
             num2str(point)];
      plot(time,vmstress(index1,:),time,yieldstrs(index1,:),…
            '--'),grid,xlabel('Time (sec)'),…
            ylabel('Effective Stress'),title(titlestr),...
            legend('VME Stress','Yield Stress');
      maxvm = max(vmstress(index1,:))
      maxys = max(yieldstrs(index1,:))
```

```
case 3,
    titlestr = ['Element #' num2str(selem) ' Integration Point #',…
                num2str(point)];
    plot(time,effstrain(index1,:)),grid,xlabel('Time (sec)'),...
       ylabel('Effective Strain'),title(titlestr);
    maxstrn = max(effstrain(index1,:))

 case 4,
    titlestr = ['Element #' num2str(selem) ' Integration Point #',…
                num2str(point)];
    plot(time,void(index1,:)),grid,xlabel('Time (sec)'),…
                title(titlestr), ylabel('Porosity');
    maxpor = max(void(index1,:))

 case 5,
    titlestr = ['Element #' num2str(selem) ' Integration Point #',…
                num2str(point)];
    plot(effplstrn(index1,:),void(index1,:)),grid,…
          xlabel('Effective Plastic Strain'),...
          ylabel('Porosity'),title(titlestr);
    maxpor = max(void(index1,:))

 case 6,
    titlestr = ['Element #' num2str(selem) ' Time: ',…
          num2str(time(tstp)) ' sec'];
    strs = zeros(ipoint,1);
    pnts = 1:ipoint;
    for p=1:ipoint
          index3 = (selem-1)*6*ipoint + (p-1)*6 + 3;
          strs(p) = stress(index3,tstp);
    end
    maxstrs = max(strs(p))
    minstrs = min(strs(p))
    plot(strs,pnts),grid,axis ij,ylabel('Integration Point'),…
          xlabel('Z Stress'), title(titlestr);

 case 7,
    titlestr = ['Element #' num2str(selem) ' Integration Point #',…
          num2str(point)];
    plot(time,stress(index+1,:)),grid,xlabel('Time (sec)'),…
          ylabel('X Stress'),title(titlestr);
    maxstrs = max(stress(index+1,:))
    minstrs = min(stress(index+1,:))

 case 8,
    titlestr = ['Element #' num2str(selem) ' Integration Point #',…
          num2str(point)];
    plot(time,stress(index+2,:)),grid,xlabel('Time (sec)'),…
          ylabel('Y Stress'),title(titlestr);
    maxstrs = max(stress(index+2,:))
    minstrs = min(stress(index+2,:))

 case 9,
    titlestr = ['Element #' num2str(selem) ' Integration Point #',…
          num2str(point)];
    plot(time,stress(index+3,:)),grid,xlabel('Time (sec)'),…
          ylabel('Z Stress'),title(titlestr);
    maxstrs = max(stress(index+3,:))
    minstrs = min(stress(index+3,:))
```

```
case 10,
   titlestr = ['Element #' num2str(selem) ' Integration Point #',…
         num2str(point)];
   subplot(2,2,1)
   plot(time,stress(index+1,:)),grid,xlabel('Time (sec)'),…
         ylabel('X Stress'),title(titlestr);
   subplot(2,2,2)
   plot(time,stress(index+2,:)),grid,xlabel('Time (sec)'),…
         ylabel('Y Stress'),title(titlestr);
   subplot(2,2,3)
   plot(time,stress(index+3,:)),grid,xlabel('Time (sec)'),…
         ylabel('Z Stress'),title(titlestr);
   subplot(2,2,4)
   plot(time,vmstress(index1,:),time,yieldstrs(index1,:),…
         '--'),grid,xlabel('Time (sec)'),…
         ylabel('Effective Stress'),…
         legend('VME Stress','Yield Stress'),title(titlestr);
   maxstrs_x = max(stress(index+1,:))
   minstrs_x = min(stress(index+1,:))
   maxstrs_y = max(stress(index+2,:))
   minstrs_y = min(stress(index+2,:))
   maxstrs_z = max(stress(index+3,:))
   minstrs_z = min(stress(index+3,:))

case 11,
   titlestr = ['Node #',num2str(snode)];
   plot(time,xu(snode,:)/dscale),grid,xlabel('Time (sec)'),…
         ylabel('X Node Displacement'),title(titlestr);
   maxdx = max(xu(snode,:))/dscale
   mindx = min(xu(snode,:))/dscale

case 12,
   titlestr = ['Node #',num2str(snode)];
   plot(time,yu(snode,:)/dscale),grid,xlabel('Time (sec)'),…
         ylabel('Y Node Displacement'),title(titlestr);
   maxdy = max(yu(snode,:))/dscale
   mindy = min(yu(snode,:))/dscale

case 13,
   titlestr = ['Node #',num2str(snode)];
   plot(time,zu(snode,:)/dscale),grid,xlabel('Time (sec)'),…
         ylabel('Z Node Displacement'),title(titlestr);
   maxdz = max(zu(snode,:))/dscale
   mindz = min(zu(snode,:))/dscale

case 14,
   titlestr = ['Node #',num2str(snode)];
   plot(time,thxu(snode,:)),grid,xlabel('Time (sec)'),…
         ylabel('Theta X'),title(titlestr);
   maxdz = max(thxu(snode,:))
   mindz = min(thxu(snode,:))

case 15,
   titlestr = ['Node #',num2str(snode)];
   plot(time,thyu(snode,:)),grid,xlabel('Time (sec)'),…
         ylabel('Theta Y'),title(titlestr);
   maxdz = max(thyu(snode,:))
   mindz = min(thyu(snode,:))
   case 16,
      titlestr = ['Node #',num2str(snode)];
      plot(time,thzu(snode,:)),grid,xlabel('Time (sec)'),...
            ylabel('Theta Z'),title(titlestr);
      maxdz = max(thzu(snode,:))
      mindz = min(thzu(snode,:))

   case 17,
      titlestr = ['Node #',num2str(snode)];
      du = sqrt(xu(snode,:).^2 + yu(snode,:).^2 + ...
            zu(snode,:).^2)/dscale;
      maxdu = max(du)
      mindu = min(du)
      plot(time,du),grid,xlabel('Time (sec)'),...
            ylabel('Mean Node Displacement'),title(titlestr);

   case 18,
      titlestr = ['Element #' num2str(selem) ' Integration Point #',...
            num2str(point)];
      plot(time,effplstrn(index1,:)),grid,xlabel('Time (sec)'),...
               ylabel('Effective Plastic Strain'),title(titlestr);
      maxplaststrn = max(effplstrn(index1,:))

   case 19,
      titlestr = ['Element #', num2str(selem) ' Integration Point #',...
            num2str(point)];
      plot(time,estrain(index1,:)),grid,xlabel('Time (sec)'),...
         ylabel('Effective Elastic Strain'),title(titlestr);
      maxelaststrn = max(estrain(index1,:))

   otherwise,
      disp('Invalid Graph Type');
end

%%%%%%%%%%%%%%%%%%%%%%%% End of DRAWSTRESS.M %%%%%%%%%%%%%%%%%%%%%%
```



```
axvm =

 2.8563e+008

axstrn =

   0.0031
```

Figure 23. Screen Capture of FEA Preprocessor.

APPENDIX C. FILE LISTING FOR FEA SHELL ELEMENT SUBROUTINE.

```
c***********************************************************************
c*    elsh4l.f
c*    Contains the following subroutines:
c*          elsh4l - element matrices for 3-D degenerated 4-node shell
c***********************************************************************
      subroutine elsh4l (nnds,xcoord,ycoord,zcoord,nesh4,ndsh4,matsh4,
     1                  nmat,matno,eprop,ndsdof,sdisp,svel,sforce,
     2                  smass,nsdof,ishel4,stssh4,mpoint,ipoint,itime)
c***********************************************************************
c     compute and assemble element matrices for shell elements with
c     4 nodes (linear analysis)
c     3-D degenerated formulation including transverse shear and
c     transverse normal deformation
c     pressure formulation
c                                             McDermott 1999
c********************************************************************
      common /fblk/ fail,array1(125)
      dimension xcoord(nnds),ycoord(nnds),zcoord(nnds),
     1          ndsh4(nesh4+1,4),matsh4(nesh4+1),ishel4(nesh4+1),
     2          matno(nmat),eprop(nmat,40),ndsdof(nnds),
     3          sdisp(nsdof),svel(nsdof),sforce(nsdof),smass(nsdof),
     4          stssh4(mpoint,nesh4*8)
      dimension eforce(24),lnode(4),evel(24),slope(10),dstrnplst(6),
     1          emass(24),gausx(4),lindex(24),strain(10),corr(2),
     2          array2(15),estrain(6),estress(6),s(6),A(2,2),Ainv(2,2),
     3          gausy(4),weighx(4),weighy(4),shape1(3),deriv1(3),
     4          shapef(4),derivl(2,4),derivg(3,4),aj(3,3),ajinv(3,3),
     5          bmtx(6,24),tmp(8),xc(4),yc(4),zc(4),estressg(6),
     6          deri1g(3,4),d1(3),edisp(24),edispg(24),rot6t(6,6),
     7          d2(3),v1(3),v2(3),v3(3),gausz(10),weighz(10),
     8          bmtxt(24,6),delt(6),t1(3,3),t1inv(3,3),B(2),
     9          rot6(6,6),rot3(3,3),estrainp(6),eforceg(24),
     1          weighxh(4),weighyh(4),weighzh(4),eforceh(24),
     2          gausxh(4),gausyh(4),gauszh(4),iplane(5)
c
      data ngausx,ngausy,ndof,n /1, 1, 6, 4/
      data ngausxh,ngausyh,ngauszh,hour1 /2, 2, 1, 0.05/
      data ref,eps /0.0,1.0e-4 /
      data delt /1.0, 1.0, 1.0, 0.0, 0.0, 0.0/
      data iplane /1, 2, 4, 5, 6/

      neldof=ndof*n
      ngausz=ipoint
      ngauss=ngausx*ngausy*ngausz
      ngaussh=ngausxh*ngausyh*ngauszh
c
c *** extract gauss points and weights for numerical integration
c
      call gauss (ngausx,gausx,weighx)
      call gauss (ngausy,gausy,weighy)
      call gauss (ngausz,gausz,weighz)
      call gauss (ngausxh,gausxh,weighxh)
      call gauss (ngausyh,gausyh,weighyh)
      call gauss (ngauszh,gauszh,weighzh)
```

```
c
c *** loop for number of elements
c
      do 370 ie=1,nesh4
c
            igausp=0
c
c *** initialize element internal force vector
c
            do 10 i=1,neldof
             eforceh(i) = 0.0
             eforce(i) = 0.0
  10        continue
c
c *** extract material property index and material properties
c
            imt=matsh4(ie)
            mattyp=matno(imt)
            densty=eprop(imt,1)
            elast = eprop(imt,2)
            poiss = eprop(imt,3)
            thick = eprop(imt,4)
      Syp = eprop(imt,5)
      q1 = eprop(imt,6)
      q2 = eprop(imt,7)
      q3 = eprop(imt,8)
      fn = eprop(imt,9)
      en = eprop(imt,10)
      sn = eprop(imt,11)
      Phi0 = eprop(imt,12)
      iseg = eprop(imt,13) + 0.1
      ystrain = Syp / elast
      strain(1) = ystrain
            if (iseg.gt.0) then
            do 20 i=1,iseg
                  slope(i) = eprop(imt,12 + i * 2)
                  strain(i+1) = eprop(imt,13 + i * 2)
   20       continue
      endif
            bulk = elast / (3.0 * (1.0 - 2.0 * poiss))
            shear = elast / (2.0 * (1.0 + poiss))
c
c *** Default Values
c
      if (q1 .ne. 0.0) then
            if (q2 .eq. 0.0) q2 = 1.0
            if (q3 .eq. 0.0) q3 = q1**2
            if (fn .ne. 0.0) then
                  if (sn .eq. 0.0) sn = 0.1
                  if (en .eq. 0.0) en = 0.3
            endif
      endif
c
c *** find dof index of each element
c
      do 30 i=1,n
            lnode(i)=ndsh4(ie,i)
            do 30 j=1,ndof
                  lindex((i-1)*ndof+j)=ndsdof(lnode(i))+j-1
   30       continue
```

```
c
c *** load displacement for element
c
            do 40 i=1,neldof
            edispg(i) = sdisp(lindex(i))
            evel(i) = svel(lindex(i))
   40       continue
c
c *** extract element nodal coordinate values
c
            do 50 i=1,n
            xc(i)=xcoord(lnode(i))
            yc(i)=ycoord(lnode(i))
            zc(i)=zcoord(lnode(i))
   50       continue
c
c *** compute direction vectors
c *** (v1 & v2: tangent, v3: normal to surface)
c
            d1(1)=xc(2)-xc(1)
            d1(2)=yc(2)-yc(1)
            d1(3)=zc(2)-zc(1)
            d2(1)=xc(4)-xc(1)
            d2(2)=yc(4)-yc(1)
            d2(3)=zc(4)-zc(1)
c
c *** Calculation of Hourglass Control Multiplier
c
c     Need Surface Area of Element
      al1=sqrt(d1(1)**2+d1(2)**2+d1(3)**2)
      al4 = sqrt(d2(1)**2+d2(2)**2+d2(3)**2)
      al2 = sqrt((xc(3)-xc(2))**2+(yc(3)-yc(2))**2+(zc(3)-zc(2))**2)
      al3 = sqrt((xc(3)-xc(4))**2+(yc(3)-yc(4))**2+(zc(3)-zc(4))**2)
      adot1 = (xc(1)-xc(2))*(xc(3)-xc(2))+(yc(1)-yc(2))*(yc(3)-yc(2))
    1 + (zc(1)-zc(2))*(zc(3)-zc(2))
      adot2 = (xc(1)-xc(4))*(xc(3)-xc(4))+(yc(1)-yc(4))*(yc(3)-yc(4))
    1 + (zc(1)-zc(4))*(zc(3)-zc(4))
      area = 0.5*(sqrt(al1**2*al2**2-adot1**2)+
    1 sqrt(al3**2*al4**2-adot2**2))
      hour = hour1 * thick**2 / area
c
c *** v3 is normal to plane
            v3(1)=d1(2)*d2(3)-d1(3)*d2(2)
            v3(2)=d1(3)*d2(1)-d1(1)*d2(3)
            v3(3)=d1(1)*d2(2)-d1(2)*d2(1)
            sum=sqrt(v3(1)**2+v3(2)**2+v3(3)**2)
            v3(1)=v3(1)/sum
            v3(2)=v3(2)/sum
            v3(3)=v3(3)/sum
c
c *** v2 is unit vector in direction of node 1 to node 4
            v2(1) = d2(1)/al4
            v2(2) = d2(2)/al4
            v2(3) = d2(3)/al4
c
c *** v1 is orthogonal to v2, in plane normal to v3
            v1(1)=v3(2)*v2(3)-v3(3)*v2(2)
            v1(2)=v3(3)*v2(1)-v3(1)*v2(3)
            v1(3)=v3(1)*v2(2)-v3(2)*v2(1)
            sum = sqrt(v1(1)**2+v1(2)**2+v1(3)**2)
            v1(1)=-v1(1)/sum
            v1(2)=-v1(2)/sum
            v1(3)=-v1(3)/sum
c
c *** Strain Transformation Matrix
            rot6(1,1)=v1(1)**2
            rot6(1,2)=v1(2)**2
            rot6(1,3)=v1(3)**2
            rot6(1,4)=v1(1)*v1(2)
            rot6(1,5)=v1(2)*v1(3)
            rot6(1,6)=v1(1)*v1(3)
            rot6(2,1)=v2(1)**2
            rot6(2,2)=v2(2)**2
            rot6(2,3)=v2(3)**2
            rot6(2,4)=v2(1)*v2(2)
            rot6(2,5)=v2(2)*v2(3)
            rot6(2,6)=v2(1)*v2(3)
            rot6(3,1)=v3(1)**2
            rot6(3,2)=v3(2)**2
            rot6(3,3)=v3(3)**2
            rot6(3,4)=v3(1)*v3(2)
            rot6(3,5)=v3(2)*v3(3)
            rot6(3,6)=v3(1)*v3(3)
            rot6(4,1)=2.0*v1(1)*v2(1)
            rot6(4,2)=2.0*v1(2)*v2(2)
            rot6(4,3)=2.0*v1(3)*v2(3)
            rot6(4,4)=v1(1)*v2(2)+v2(1)*v1(2)
            rot6(4,5)=v1(2)*v2(3)+v2(2)*v1(3)
            rot6(4,6)=v1(3)*v2(1)+v2(3)*v1(1)
            rot6(5,1)=2.0*v2(1)*v3(1)
            rot6(5,2)=2.0*v2(2)*v3(2)
            rot6(5,3)=2.0*v2(3)*v3(3)
            rot6(5,4)=v2(1)*v3(2)+v3(1)*v2(2)
            rot6(5,5)=v2(2)*v3(3)+v3(2)*v2(3)
            rot6(5,6)=v2(3)*v3(1)+v3(3)*v2(1)
            rot6(6,1)=2.0*v3(1)*v1(1)
            rot6(6,2)=2.0*v3(2)*v1(2)
            rot6(6,3)=2.0*v3(3)*v1(3)
            rot6(6,4)=v3(1)*v1(2)+v1(1)*v3(2)
            rot6(6,5)=v3(2)*v1(3)+v1(2)*v3(3)
            rot6(6,6)=v3(3)*v1(1)+v1(3)*v3(1)
c
c *** Transpose of Transformation Matrix
c
            do 60 i=1,6
            do 60 j=1,6
                  rot6t(i,j)=rot6(j,i)
   60       continue
c
c *** Rotation tranformation matrix, and its inverse
c
            do 70 i=1,3
            t1(i,1) = v1(i)
            t1(i,2) = v2(i)
            t1(i,3) = v3(i)
   70       continue
            call inv3x3(t1,t1inv,det)
c
c *** Transform Rotation displacement to local coordinates
c
```

```
            do 80 i=1,4
            do 80 j=1,3
                  edisp((i-1)*6+j)=edispg((i-1)*6+j)
                  edisp((i-1)*6+3+j)=edispg((i-1)*6+4)*t1inv(j,1)
     1                        +edispg((i-1)*6+5)*t1inv(j,2)
     2                             +edispg((i-1)*6+6)*t1inv(j,3)
   80       continue
c
c *** Begining of Hourglass control
c
      if (hour.gt.0.0) then
       do 140 ix=1,ngausxh
        rc=gausxh(ix)
        wx=weighxh(ix)
         do 140 iy=1,ngausyh
          sc=gausyh(iy)
          wy=weighyh(iy)
           do 140 iz=1,ngauszh
            tc=gauszh(iz)
            wz=weighzh(iz)
c
c *** compute shape functions and derivatives
c
            call shp2d4 (rc,sc,shapef)
            call der2d4 (rc,sc,derivl)
            call shp1d2 (tc,shape1)
            call der1d2 (tc,derivl)
c
c *** compute jacobian matrix and its inverse
c
            hz=thick*0.5*(shape1(2)*(1.0-ref)-shape1(1)*(1.0+ref))
            hzdt=thick*0.5

      aj(1,1)=derivl(1,1)*xc(1)+derivl(1,2)*xc(2)+derivl(1,3)*xc(3)+
     1                  derivl(1,4)*xc(4)+derivl(1,1)*hz*v3(1)+
     2                  derivl(1,2)*hz*v3(1)+derivl(1,3)*hz*v3(1)+
     3                  derivl(1,4)*hz*v3(1)

      aj(2,1)=derivl(2,1)*xc(1)+derivl(2,2)*xc(2)+derivl(2,3)*xc(3)+
     1                  derivl(2,4)*xc(4)+derivl(2,1)*hz*v3(1)+
     2                  derivl(2,2)*hz*v3(1)+derivl(2,3)*hz*v3(1)+
     3                  derivl(2,4)*hz*v3(1)
            aj(3,1)=hzdt*v3(1)

      aj(1,2)=derivl(1,1)*yc(1)+derivl(1,2)*yc(2)+derivl(1,3)*yc(3)+
     1                  derivl(1,4)*yc(4)+derivl(1,1)*hz*v3(2)+
     2                  derivl(1,2)*hz*v3(2)+derivl(1,3)*hz*v3(2)+
     3                  derivl(1,4)*hz*v3(2)

      aj(2,2)=derivl(2,1)*yc(1)+derivl(2,2)*yc(2)+derivl(2,3)*yc(3)+
     1                  derivl(2,4)*yc(4)+derivl(2,1)*hz*v3(2)+
     2                  derivl(2,2)*hz*v3(2)+derivl(2,3)*hz*v3(2)+
     3                  derivl(2,4)*hz*v3(2)
            aj(3,2)=hzdt*v3(2)

      aj(1,3)=derivl(1,1)*zc(1)+derivl(1,2)*zc(2)+derivl(1,3)*zc(3)+
     1                  derivl(1,4)*zc(4)+derivl(1,1)*hz*v3(3)+
     2                  derivl(1,2)*hz*v3(3)+derivl(1,3)*hz*v3(3)+
     3                  derivl(1,4)*hz*v3(3)
```

```
      aj(2,3)=derivl(2,1)*zc(1)+derivl(2,2)*zc(2)+derivl(2,3)*zc(3)+
     1                  derivl(2,4)*zc(4)+derivl(2,1)*hz*v3(3)+
     2                  derivl(2,2)*hz*v3(3)+derivl(2,3)*hz*v3(3)+
     3                  derivl(2,4)*hz*v3(3)
            aj(3,3)=hzdt*v3(3)
c
            call inv3x3 (aj,ajinv,det)
c
c *** compute global derivatives and strain-nodal displacement matrix
c
            do 90 i=1,n

      derivg(1,i)=ajinv(1,1)*derivl(1,i)+ajinv(1,2)*derivl(2,i)
      derivg(2,i)=ajinv(2,1)*derivl(1,i)+ajinv(2,2)*derivl(2,i)
      derivg(3,i)=ajinv(3,1)*derivl(1,i)+ajinv(3,2)*derivl(2,i)
                  deri1g(1,i)=ajinv(1,3)*hzdt
                  deri1g(2,i)=ajinv(2,3)*hzdt
                  deri1g(3,i)=ajinv(3,3)*hzdt
   90       continue
c
            do 100 i=1,6
                  do 100 j=1,24
                         bmtx(i,j)=0.0
  100       continue
c
            do 110 i=1,n
                  i1=(i-1)*6+1
                  i2=i1+1
                  i3=i2+1
                  i4=i3+1
                  i5=i4+1
                  i6=i5+1
                  gk1=derivg(1,i)*hz+shapef(i)*deri1g(1,i)
                  gk2=derivg(2,i)*hz+shapef(i)*deri1g(2,i)
                  gk3=derivg(3,i)*hz+shapef(i)*deri1g(3,i)
c
                  bmtx(1,i1)=derivg(1,i)
                  bmtx(1,i4)=gk1*(-v2(1))
                  bmtx(1,i5)=gk1*v1(1)
                  bmtx(1,i6)=gk1*v3(1)
                  bmtx(2,i2)=derivg(2,i)
                  bmtx(2,i4)=gk2*(-v2(2))
                  bmtx(2,i5)=gk2*v1(2)
                  bmtx(2,i6)=gk2*v3(2)
                  bmtx(3,i3)=derivg(3,i)
                  bmtx(3,i4)=gk3*(-v2(3))
                  bmtx(3,i5)=gk3*v1(3)
                  bmtx(3,i6)=gk3*v3(3)
                  bmtx(4,i1)=derivg(2,i)
                  bmtx(4,i2)=derivg(1,i)
                  bmtx(4,i4)=gk2*(-v2(1))+gk1*(-v2(2))
                  bmtx(4,i5)=gk2*v1(1)+gk1*v1(2)
                  bmtx(4,i6)=gk2*v3(1)+gk1*v3(2)
                  bmtx(5,i2)=derivg(3,i)
                  bmtx(5,i3)=derivg(2,i)
                  bmtx(5,i4)=gk3*(-v2(2))+gk2*(-v2(3))
                  bmtx(5,i5)=gk3*v1(2)+gk2*v1(3)
                  bmtx(5,i6)=gk3*v3(2)+gk2*v3(3)
                  bmtx(6,i1)=derivg(3,i)
                  bmtx(6,i3)=derivg(1,i)
                  bmtx(6,i4)=gk3*(-v2(1))+gk1*(-v2(3))
                  bmtx(6,i5)=gk3*v1(1)+gk1*v1(3)
                  bmtx(6,i6)=gk3*v3(1)+gk1*v3(3)
  110       continue
c
                  do 120 i=1,6
                  do 120 j=1,neldof
  120                   bmtxt(j,i)=bmtx(i,j)
c
            weight = wx * wy * wz
            detwt = det * weight
c
c *** calculate strain
c
            call mmult(6,24,1,bmtx,edisp,estrainp,0,1.0)
c
c *** Transform strain to local coordinates system
c
            call mmult(6,6,1,rot6,estrainp,estrain,0,1.0)
c
c *** Subtract any prior plastic strain
c
            do 130 i=1,6
  130             estrain(i) = estrain(i) - stssh4(12+i,issts)
c
c *** Calculate stress using plane-strain formulas
c
      estress(1) = elast/(1.0-poiss**2)*(estrain(1) + poiss*estrain(2))
      estress(2) = elast/(1.0-poiss**2)*(poiss*estrain(1) + estrain(2))
                  estress(3) = elast * estrain(3)
                  estress(4) = shear * estrain(4)
                  estress(5) = 5.0 /6.0 * shear * estrain(5)
                  estress(6) = 5.0 /6.0 * shear * estrain(6)
c
c *** Transform stress back to global coordinates
c
            call mmult(6,6,1,rot6t,estress,estressg,0,1.0)
c
c *** find eforceh for this gauss point and sum to element force
c
            call mmult(24,6,1,bmtxt,estressg,eforceh,1,detwt)
c
  140       continue
            endif
c
c *** end of hourglass loop
c
c
c *** numerical integration loop for membrane
c
            icount=0
            do 300 ix=1,ngausx
             rc=gausx(ix)
             wx=weighx(ix)
             do 300 iy=1,ngausy
              sc=gausy(iy)
              wy=weighy(iy)
              do 300 iz=1,ngausz
               tc=gausz(iz)
               wz=weighz(iz)
c
c *** increase counter
c
            igausp=igausp+1
            isssts=(ie-1)*8+igausp
c
c *** Initialize stssh4 if first time
c
            if (itime .eq. 1) then
                  do 150 i=13,21
  150                    stssh4(i,issts) = 0.0
c
                  stssh4(19,issts) = Syp
                  stssh4(20,issts) = Phi0
            endif
c
c *** compute shape functions and derivatives
c
                  call shp2d4 (rc,sc,shapef)
                  call der2d4 (rc,sc,derivl)
                  call shp1d2 (tc,shape1)
                  call der1d2 (tc,derivl)
c
c *** compute jacobian matrix and its inverse
c
                  hz=thick*0.5*(shape1(2)*(1.0-ref)-shape1(1)*(1.0+ref))
                  hzdt=thick*0.5

      aj(1,1)=derivl(1,1)*xc(1)+derivl(1,2)*xc(2)+derivl(1,3)*xc(3)+
     1                  derivl(1,4)*xc(4)+derivl(1,1)*hz*v3(1)+
     2                  derivl(1,2)*hz*v3(1)+derivl(1,3)*hz*v3(1)+
     3                  derivl(1,4)*hz*v3(1)

      aj(2,1)=derivl(2,1)*xc(1)+derivl(2,2)*xc(2)+derivl(2,3)*xc(3)+
     1                  derivl(2,4)*xc(4)+derivl(2,1)*hz*v3(1)+
     2                  derivl(2,2)*hz*v3(1)+derivl(2,3)*hz*v3(1)+
     3                  derivl(2,4)*hz*v3(1)
                  aj(3,1)=hzdt*v3(1)

      aj(1,2)=derivl(1,1)*yc(1)+derivl(1,2)*yc(2)+derivl(1,3)*yc(3)+
     1                  derivl(1,4)*yc(4)+derivl(1,1)*hz*v3(2)+
     2                  derivl(1,2)*hz*v3(2)+derivl(1,3)*hz*v3(2)+
     3                  derivl(1,4)*hz*v3(2)

      aj(2,2)=derivl(2,1)*yc(1)+derivl(2,2)*yc(2)+derivl(2,3)*yc(3)+
     1                  derivl(2,4)*yc(4)+derivl(2,1)*hz*v3(2)+
     2                  derivl(2,2)*hz*v3(2)+derivl(2,3)*hz*v3(2)+
     3                  derivl(2,4)*hz*v3(2)
                  aj(3,2)=hzdt*v3(2)

      aj(1,3)=derivl(1,1)*zc(1)+derivl(1,2)*zc(2)+derivl(1,3)*zc(3)+
     1                  derivl(1,4)*zc(4)+derivl(1,1)*hz*v3(3)+
     2                  derivl(1,2)*hz*v3(3)+derivl(1,3)*hz*v3(3)+
     3                  derivl(1,4)*hz*v3(3)

      aj(2,3)=derivl(2,1)*zc(1)+derivl(2,2)*zc(2)+derivl(2,3)*zc(3)+
     1                  derivl(2,4)*zc(4)+derivl(2,1)*hz*v3(3)+
     2                  derivl(2,2)*hz*v3(3)+derivl(2,3)*hz*v3(3)+
     3                  derivl(2,4)*hz*v3(3)
```

```
                  aj(3,3)=hzdt*v3(3)
c
                  call inv3x3 (aj,ajinv,det)
c
c *** compute global derivatives and strain-nodal displacement matrix
c
                  do 160 i=1,n

      derivg(1,i)=ajinv(1,1)*derivl(1,i)+ajinv(1,2)*derivl(2,i)
      derivg(2,i)=ajinv(2,1)*derivl(1,i)+ajinv(2,2)*derivl(2,i)
      derivg(3,i)=ajinv(3,1)*derivl(1,i)+ajinv(3,2)*derivl(2,i)
                  deri1g(1,i)=ajinv(1,3)*hzdt
                  deri1g(2,i)=ajinv(2,3)*hzdt
                  deri1g(3,i)=ajinv(3,3)*hzdt
  160       continue
c
                  do 170 i=1,6
                  do 170 j=1,24
                        bmtx(i,j)=0.0
  170       continue
c
                  do 180 i=1,n
                  i1=(i-1)*6+1
                  i2=i1+1
                  i3=i2+1
                  i4=i3+1
                  i5=i4+1
                  i6=i5+1
                  gk1=derivg(1,i)*hz+shapef(i)*deri1g(1,i)
                  gk2=derivg(2,i)*hz+shapef(i)*deri1g(2,i)
                  gk3=derivg(3,i)*hz+shapef(i)*deri1g(3,i)
c
                  bmtx(1,i1)=derivg(1,i)
                  bmtx(1,i4)=gk1*(-v2(1))
                  bmtx(1,i5)=gk1*v1(1)
                  bmtx(1,i6)=gk1*v3(1)
                  bmtx(2,i2)=derivg(2,i)
                  bmtx(2,i4)=gk2*(-v2(2))
                  bmtx(2,i5)=gk2*v1(2)
                  bmtx(2,i6)=gk2*v3(2)
                  bmtx(3,i3)=derivg(3,i)
                  bmtx(3,i4)=gk3*(-v2(3))
                  bmtx(3,i5)=gk3*v1(3)
                  bmtx(3,i6)=gk3*v3(3)
                  bmtx(4,i1)=derivg(2,i)
                  bmtx(4,i2)=derivg(1,i)
                  bmtx(4,i4)=gk2*(-v2(1))+gk1*(-v2(2))
                  bmtx(4,i5)=gk2*v1(1)+gk1*v1(2)
                  bmtx(4,i6)=gk2*v3(1)+gk1*v3(2)
                  bmtx(5,i2)=derivg(3,i)
                  bmtx(5,i3)=derivg(2,i)
                  bmtx(5,i4)=gk3*(-v2(2))+gk2*(-v2(3))
                  bmtx(5,i5)=gk3*v1(2)+gk2*v1(3)
                  bmtx(5,i6)=gk3*v3(2)+gk2*v3(3)
                  bmtx(6,i1)=derivg(3,i)
                  bmtx(6,i3)=derivg(1,i)
                  bmtx(6,i4)=gk3*(-v2(1))+gk1*(-v2(3))
                  bmtx(6,i5)=gk3*v1(1)+gk1*v1(3)
                  bmtx(6,i6)=gk3*v3(1)+gk1*v3(3)
  180       continue
```

```
c
c *** calculate transpose of bmtx
c
                  do 190 i=1,6
                  do 190 j=1,neldof
  190                    bmtxt(j,i)=bmtx(i,j)
c
            weight = wx * wy * wz
                  detwt = det * weight
                  icount=icount+1
                  tmp(icount)=detwt
c
c *** calculate strain
c
            call mmult(6,24,1,bmtx,edisp,estrainp,0,1.0)
c
c *** Transform strain to local coordinates system
c
            call mmult(6,6,1,rot6,estrainp,estrain,0,1.0)
c
c *** Subtract any prior plastic strain
c
            do 200 i=1,6
  200             estrain(i) = estrain(i) - stssh4(12+i,issts)
c
            stress0 = stssh4(19,issts)
            Phi = stssh4(20,issts)
            plastrn = stssh4(21,issts)
c
c *** Calculate stress using plane-strain formulas
c
      estress(1) = elast/(1.0-poiss**2)*(estrain(1) + poiss*estrain(2))
      estress(2) = elast/(1.0-poiss**2)*(poiss*estrain(1) + estrain(2))
                  estress(3) = elast * estrain(3)
                  estress(4) = shear * estrain(4)
                  estress(5) = 5.0 /6.0 * shear * estrain(5)
                  estress(6) = 5.0 /6.0 * shear * estrain(6)
c
c *** Elastic-Plastic Calculations
c
            p = -(estress(1) + estress(2) + estress(3)) / 3.0
c
            do 210 i=1,6
  210                   s(i) = estress(i) + p * delt(i)
c
            q = sqrt(1.5 *(s(1)**2 + s(2)**2 + s(3)**2 + 2.0 * s(4)**2 +
      1           2.0 * s(5)**2 + 2.0 * s(6)**2))
            F = (q / stress0)**2+2.0*q1*Phi*cosh(-1.5*q2*p/stress0)-
      1           (1.0+q3* Phi**2)
c
             if (F.gt.0) then
                  iter = 0
                  dep = 0.0
                  deq = 0.0
c
c ** Begin Iteration Loop **
c
  220             c1 = 3.0*q2/(2.0*stress0**2)
                  Fq1 = 2.0 * q / (stress0**2)
                  Fq2 = 2.0 / (stress0**2)
```

```
                Fp1 = 2.0 * q1 * Phi * -c1 * sinh(-c1 * p)
                Fp2 = 2.0 * q1 * Phi * c1**2 * cosh(-c1 * p)
                Fs1 = -q **2 / (2.0 * stress0**3) + 2.0*q1*Phi*c1*
     1                p*sinh(-c1*p)
                Fsp = 2.0*q1*Phi*(-c1**2 * p*cosh(-c1*p) +
     1          c1/stress0 * sinh(-c1*p))
                Fsq = -q / stress0**3

                stress1 = Syp
                es1 = stress0/elast
                e1 = -p*dep/((1-Phi)*stress0)
                ps1 = plastrn + e1 + es1
                iflag = 0
                do 230 i=1,iseg
                      if (iflag .eq. 0) then
                       if(ps1 .lt. strain(i+1)) then
                        stress1 = stress1 + slope(i)*(ps1-strain(i))
                        iseg1 = i
                        iflag = 1
                       else
                  stress1 = stress1 + slope(i)*(strain(i+1)-strain(i))
                       endif
                      endif
230             continue
```

```
c
c  ***  Numerical Error Trap
c
                  if ((stress1 .eq. stress0) .or. (abs(e1) .eq. 0.0))
then
                        dsdep = slope(iseg1)
                  else
                        dsdep = (stress1-stress0)/e1
                  endif
                  if (abs(dsdep) .gt. elast) then
                        dsdep = slope(iseg1)
                  endif
c
                  stress1 = Syp
                  e1 = q * deq / ((1-Phi) * stress0)
                  ps1 = plastrn + e1
                  iflag = 0
                  do 240 i=1,iseg
                   if (iflag .eq. 0) then
                    if(ps1 .lt. strain(i+1)) then
                     stress1 = stress1 + slope(i)*(ps1-strain(i))
                     iseg1 = i
                     iflag = 1
                    else
                     stress1 = stress1 + slope(i)*(strain(i+1)-
strain(i))
                    endif
                   endif
  240             continue
c
c *** Numerical Error Trap
c
                  if ((stress1 .eq. stress0).or.(abs(e1) .eq. 0.0)) then
                        dsdeq = slope(iseg1)
                  else
                        dsdeq = (stress1-stress0)/e1
                  endif
                  if (abs(dsdeq) .gt. elast) then
                        dsdeq = slope(iseg1)
                  endif
c
c *** Using only stress0 derivatives
c
                  A(1,1) = bulk * Fp1+Fs1*dsdep
                  A(1,2) = -3.0*shear*Fq1+Fs1*dsdeq
                  A(2,1) = Fq1+deq*(bulk*Fp2+Fsp*dsdep)+dep*Fsq*dsdep
            A(2,2) = Fp1+dep*(-3.0*shear*Fq2+Fsq*dsdeq)+deq*Fsp*dsdeq
c
                  B(1) = -1.0 * F
                  B(2) = -1.0 * dep * Fq1 - deq * Fp1
c
                  call inv2x2(A,Ainv,d)
c
                  call mmult(2,2,1,Ainv,B,corr,0,1.0)
c
                  Phi = stssh4(20,issts)
                  plastrn = stssh4(21,issts)
                  stress0 = stssh4(19,issts)
c
                  dep = dep + corr(1)
                  deq = deq + corr(2)
c
                  degrowth = 0.0
                  do 250 i=1,5
                        ind = iplane(i)
                        dstrnplst(ind) = dep * delt(ind) / 3.0 +
     1                  deq * 3.0 * s(ind) / (2.0 * q)
  250             continue
c
                  dstrnplst(3) = dep / 3.0 + deq * estress(3) / q
c
                  do 260 i=1,3
  260                   degrowth = degrowth + dstrnplst(i)
c
c ** Calculate New stress, p, q, and F
c
      estress(1) = elast/(1.0-poiss**2)*(estrain(1) - dstrnplst(1) +
    1              poiss*(estrain(2)-dstrnplst(2)))
                   estress(2) = elast/(1.0-poiss**2)*
     1         (poiss*(estrain(1)-dstrnplst(1))+estrain(2)-dstrnplst(2))
                  estress(3) = elast * (estrain(3)-dstrnplst(3))
                  estress(4) = shear * (estrain(4)-dstrnplst(4))
            estress(5) = 5.0 /6.0 * shear * (estrain(5)-dstrnplst(5))
            estress(6) = 5.0 /6.0 * shear * (estrain(6)-dstrnplst(6))
                  p = (estress(1) + estress(2) + estress(3)) / -3.0
c
                  do 270 i=1,6
  270                   s(i) = estress(i) + p * delt(i)
c
                  deltep = (q*deq - p * dep)/((1-Phi)*stress0)
c
c Prevent reduction in plastic strain
                  if (deltep .lt. 0.0) then
                        deltep = 0.0
                  endif
c
                  plastrn = plastrn + deltep
                  if (fn .ne. 0.0) then
             anucl = fn/(sn*2.50662827463)*exp(-.5*((plastrn-en)/sn)**2)
                  else
                   anucl = 0.0
                  endif
c
                  Phi1 = Phi + (1 - Phi) * degrowth + anucl * deltep
c
c Phi cannot decrease
                  if (Phi1 .gt. Phi) Phi = Phi1
c
c Turn off Void Effects if q1 = 0.0
                  if (q1 .eq. 0.0) Phi = 0.0
c
c Calculate Strain Hardening
                  elastrn = stress0/elast
                  stress0 = Syp
                  iflag = 0
                  do 280 i=1,iseg
                        if (iflag .eq. 0) then
                         ps1 = plastrn + elastrn
                         if(ps1 .lt. strain(i+1)) then
                          stress0 = stress0 + slope(i)*(ps1-strain(i))
                          iflag = 1
                         else
                    stress0 = stress0 + slope(i)*(strain(i+1)-strain(i))
                         endif
                        endif
  280             continue
c
            q = sqrt(1.5 *(s(1)**2 + s(2)**2 + s(3)**2 + 2.0 * s(4)**2 +
      1                 2.0 * s(5)**2 + 2.0 * s(6)**2))
c
            F = (q / stress0)**2+2.0*q1*Phi*cosh(-1.5*q2*p/stress0)-
      1                 (1.0+q3* Phi**2)
c
                  iter = iter + 1
c
                  if (iter.gt.500) then
                        write(*,*) 'Failed to Converge'
                        stop
                  endif
                  if(abs(F).gt.eps) goto 220
c
            endif
c *** End of Plastic Deformation Section
c
c
c *** Transform stress back to global coordinates
c
            call mmult(6,6,1,rot6t,estress,estressg,0,1.0)
c
c *** find eforce for this gauss point and sum to element force
c
            call mmult(24,6,1,bmtxt,estressg,eforce,1,detwt)
c
c *** save stress and strain
c
            do 290 i = 1,6
                  stssh4(i,issts)=estress(i)
                  stssh4(i+6,issts)=estrain(i) - dstrnplst(i)
                  stssh4(i+12,issts)=stssh4(i+12,issts) + dstrnplst(i)
  290       continue
c
            stssh4(19,issts) = stress0
            stssh4(20,issts) = Phi
            stssh4(21,issts) = plastrn
c
c *** end of integration loop
c
  300       continue
c
c *** Hourglass Control Correction
c
      do 310 i = 1,24
  310       eforce(i) = eforce(i) + hour * (eforceh(i) - eforce(i))
c
c *** Transform Moments back to Global Coordinate system
c
            do 320 i=1,4
            do 320 j=1,3
                        eforceg((i-1)*6+j)=eforce((i-1)*6+j)
                  eforceg((i-1)*6+3+j)=eforce((i-1)*6+4)*t1(j,1)
```

```
     1                              +eforce((i-1)*6+5)*t1(j,2)
     2                              +eforce((i-1)*6+6)*t1(j,3)
  320       continue
c
c *** compute lumped mass matrix
c
            sum=0.0
            do 340 i=1, ngauss
  340       sum=sum+tmp(i)
c
       ems=sum/n
c
            do 350 i=1, neldof
  350       emass(i)=ems*densty
c
c *** assemble mass and force matrices into system matrix
c
            do 360 i=1, neldof
            sforce(lindex(i)) = sforce(lindex(i)) + eforceg(i)
            smass(lindex(i)) = smass(lindex(i)) + emass(i)
  360       continue
c
c *** end of element loop
c
  370 continue
c
         return
         end
```

LIST OF REFERENCES

[1] B. L. Koziey and F. A. Mirza, "Consistent Thick Shell Element," *Comput. Struct.* **65.4**, 531-549 (1997).

[2] D. J. Allman, Evaluation of the Constant Strain Triangle with Drilling Rotations, *Int. J. Numer. Methods Engrg.* **26**, 2645-2655 (1988).

[3] A. M. Khaski and A. L Soler, "A Finite Element for General Thick Walled Shell Structures," *J. Press. Vessel Tech.* **107**, 126-133 (1985).

[4] R. D. Cook, "Four-Node 'Flat' Shell Element: Drilling Degrees of Freedom, Membrane-Bending Coupling, Warped Geometry, and Behavior," *Comput. Struct.* **50.4**, 549-555 (1994).

[5] B. L. Kemp, C. Cho, and S. W. Lee, "A Four-Node Solid Shell Element Formulation with Assumed Strain," *Int. J. Numer. Methods Engrg.* **43**, 909-924 (1998).

[6] Y. Zhu and T. Zacharia, "A New One-Point Quadrature, Quadrilateral Shell Element with Drilling Degrees of Freedom," *Comp. Methods Appl. Mech. Engrg.* **136**, 165-203 (1996).

[7] Q. Zeng and A. Comescure, "A New One-Point Quadrature, General Non-linear Quadrilateral Shell Element with Physical Stabilization," *Int. J. Numer. Methods Engrg.* **42**, 1307-1338 (1998).

[8] G. Rengarajan, M. A. Aminpour, and N. F. Knight, Jr., "Improved Assumed-Stress Hybrid Shell Element with Drilling Degrees of Freedom for Linear Stress, Buckling and Free Vibration Analysis," *Int. J. Numer. Methods Engrg.* **38**, 1917-1943 (1995).

[9] D. Briassoulis, "The C^0 Shell Plate and Beam Elements Freed from their Deficiencies," *Comp. Methods Appl. Mech. Engrg.* **72**, 243-266 (1989).

[10] A. Ibrahimbegovic, R. L. Taylor, and E. L. Wilson, "A Robust Quadrilateral Membrane Finite Element with Drilling Degrees of Freedom," *Int. J. Numer. Methods Engrg.* **30**, 445-457 (1990).

[11] S. W. Key and C. C. Hoff, "An Improved Constant Membrane and Bending Stress Shell Element for Explicit Transient Dynamics," *Comp. Methods Appl. Mech. Engrg.* **124**, 33-47 (1995).

[12] K. S. Lay, "Shell Finite Element Formulated on Shell Middle Surface," *J. Engrg Mech.* **119.10**, 1973-1992 (1993).

[13] F. L. Addessio, J. N. Johnson, and P. J. Maudlin, "The Effect of Void Growth on Taylor Cylinder Impact Experiments," *J. Appl. Physics* **73.11**, 7288-7297 (1993).

[14] J. W. Hancock and R. D. Thomson, "Strain and Stress Concentrations in Ductile Fracture by Void Nucleation, Growth and Coalescence," *Mat. Sci. Tech.* **1**, 684-690 (1985).

[15] C. C. Chu and A. Needleman, "Void Nucleation Effects in Biaxially Stretched Sheets," *J. Engrg. Mat. Tech.* **102**, 249-256 (1980).

[16] V. Tvergaard, "Influence of Voids on Shear Band Instabilities under Plane Strain Conditions," *Int. J. Fracture* **17.4**, 389-406 (1981).

[17] A. L. Gurson, "Continuum Thoery of Ductile Rupture by Void Nucleation and Growth: Part I – Yield Criteria and Flow Rules for Porous Ductile Materials," *J. Engrg. Mat. Tech.* **99**, 2-15 (1977).

[18] J. C. Nagtegaal, D. M. Parks, and J. R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Comput. Methods Appl. Mech. Engrg.* **4**, 153-177 (1974).

[19] G. Shi and G. Z. Voyiadjis, "A Computational Model for FE Ductile Plastic Damage Analysis and Plate Bending," *J. Appl. Mech.* **60**, 749-758 (1993).

[20] J. H. Lee and Y. Zhang, "A Finite-Element Work-Hardening Plasticity Model of the Uniaxial Compression and Subsequent Failure of Porous Cylinders Including Effects of Void Nucleation and Growth – Part I: Plastic Flow and Damage," *J. Engrg. Mat. Tech.* **116**, 69-79 (1994).

[21] N. Aravas, "On the Numerical Integration of a Class of Pressure-Dependent Plasticity Models," *Int. J. Numer. Methods Engrg.* **24**, 1395-1416 (1987).

[22] T. J. R. Hughes and F. Brezzi, "On Drilling Degrees of Freedom," *Comput. Methods Appl. Mech. Engrg.* **72**, 105-121 (1989).

[23] F. Essenburg, "On the Significance of the Inclusion of the Effect of Transverse Normal Strain in Problems Involving Beams With Surface Constraints," *J. Appl. Mech.* 127-132 (March 1975).

[24] R. D. Cook, *Concepts and Applications of Finite Element Analysis*, 2nd ed., Wiley, New York, 1981.

[25] T. Belytschko, C. S. Tsay, and W. K. Liu, "A Stabilization Matrix for the Bilinear Mindlin Plate Element," *Comput. Methods Appl. Mech. Engrg.* **29**, 313-327 (1981).

[26] R. H. MacNeal and R. L Harder, "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Design and Analysis*, ed. W. D. Pilkey, Vol. 1, North-Holland, Amsterdam, 3-20 (1985).

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Professor Young W. Kwon, Code ME/Kw 2
   Naval Postgraduate School
   Monterey, California 93943

4. LT Patrick M. McDermott, USN 4
   80 La Havre Circle
   Florissant, MO 63031

5. Thomas E. McDermott, Ph.D., PE 1
   1800 Jefferson Ridge Dr.
   Jefferson Hills, PA 15025-3431